NEURAL NETWORK CONTROL OF AUTOCLAVE CURING OF COMPOSITE MATERIALS

Thesis Submitted to

The School of Engineering

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Chemical Engineering

by

Maria Claudia Baptista

UNIVERSITY OF DAYTON

Dayton, Ohio

December, 1994

ABSTRACT

NEURAL NETWORK CONTROL OF AUTOCLAVE CURING OF COMPOSITE MATERIALS

Baptista, Maria Claudia
University of Dayton, 1994

Advisor: Dr. C. William Lee

A back propagation neural network has been developed to determine the temperature cure cycle of a fiber reinforced epoxy matrix composite material in an autoclave. The self-directed neural network controller performed temperature control by adjusting the set-point of the autoclave based on five sensor inputs. The curing process was simulated by a one-dimensional heat transfer model that included heat source and convective boundary conditions. These two programs, the simulator and the controller, were installed on two separate computers. The neural network controller was developed using NeuroWindows™ in the Visual Basic™ environment.

The neural network controller used for testing consists of an input layer with five neurons that represent the process and material states, one hidden layer with six neurons, and an output layer with a single neuron for temperature set point adjustment. The neural network controller, when trained by a cure cycle for a given thickness panel, was able to

NEURAL NETWORK CONTROL OF AUTOCLAVE CURING OF COMPOSITE MATERIALS

Approved by:

C. William Lee, Ph.D.
Professor
Committee Chairperson

Tony E. Saliba, Ph.D.
Associate Professor
Committee Member

Kevin J. Myers, D.Sc., P.E.
Associate Professor
Committee Member

Donald L. Moon, Ph.D.
Associate Dean
Graduate Engineering Programs & Research
School of Engineering

Joseph Lestingi, D.Eng., P.E.
Dean
School of Engineering

cure the same thickness panel. This was verified by two cases. When a network was trained by a cycle for a thin panel, it was unable to satisfactorily control the cure of a thick panel, and vice versa. When the two training sets are combined to train a network, the resulting controller was unable to interpolate to control the cure of a panel of an intermediate thickness.

It is surmised that the back propagation neural network is capable of capturing an operator's decision mechanism on temperature control of a heat transfer process as it was able to carry out a cure cycle having been given only randomized instances of the process state. The five process variables chosen as the input vector can only represent a particular case of heat transfer process and are not enough for the network to generalize.

ACKNOWLEDGMENTS

I wish to acknowledge and express my appreciation to the members of the committee, Dr. Kevin J. Myers and Dr. Tony E. Saliba. My gratitude goes also to Dr. Ronald A. Servais and Dr. Sarwan Sandhu for their encouragement. In addition, special thanks are due to my advisor, Dr. C. William Lee for his precious guidance and patience during this work.

I thank Fulbright Commission, Encyclopedia Britannica, American Chamber of Commerce in Sao Paulo, University of Dayton Department of Chemical and Materials Engineering, and Rhodia S.A. for the financial support provided during my master's program at the University of Dayton.

I also would like to thank my family and my many friends for their help and encouragement. And finally to my grandparents, Adelia and Benedicto Gomes, to whom this work is dedicated.

PREFACE

Artificial Intelligence has stimulated the interest of researchers in various fields of knowledge. Some are very interested in finding the analogy with the physiology of physical systems, in understanding how human brain processes information, how learning occurs, where the information is stored and how memory works. There are so many questions surrounding the field and very few answers.

The development of simplified mathematical models also interested engineers looking for new tools to solve practical problems. The interest in applying Artificial Intelligence to solve Chemical Engineering problems has increased substantially lately. Some problems in the Chemical Process Industries cannot be solved by standard mathematical techniques or their solutions can be complicated, costly, and time consuming. In order to solve such problems, alternative methods have been developed. One of the proposed methods is Artificial Neural Networks. It can be applied in problems where the knowledge about the process is available but there is no mathematical model to represent it. An application of neural networks to process control is presented in this work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

C_p	composite heat capacity, J kg^{-1} K^{-1}
ΔH	resin heat of reaction, J kg^{-1}
E_p	RMS error between neural network calculated output and desired output
F	activation function
h	heat transfer coefficient, J m^{-2} K^{-1} s^{-1}
J	error function, sum of RMS errors
k	composite thermal conductivity, J m^{-1} K^{-1} s^{-1}
r	reaction rate, s^{-1}
S	weighted sum of neuron inputs
T	local temperature, K
t	time, s
w	resin mass fraction, dimensionless
w_i	weighting factor for input to a neuron
x	position, m
x_i	inputs to a neuron
y	neuron network output
y_c	neural network calculated output

y_d	desired output
$\Delta_p w_{ji}$	change in the weight from node i to node j at pattern p
ρ	composite density, kg m^{-3}
η	gain, or learning rate
α	momentum
δ_{pj}	error signal, distance from the activation level of node j to its desired level

CHAPTER I

INTRODUCTION

The importance of composite materials has increased substantially in various segments of industry. In the last decade the production of composites grew 40 percent from 0.87 to 1.22 million tons in the U.S. [44]. Advanced composite materials are basically a combination of a fiber phase and a matrix phase. They can be used in a wide variety of applications. There are applications in the aircraft industry, in commercial areas as diverse as sports and recreational equipment, as well as in automotive and construction industries. With such diverse applications, it is necessary to produce parts with different shapes and properties. As a consequence, the manufacturing process used for a given application has to be adapted in order to obtain a final product with the required properties.

In an epoxy resin (matrix) / graphite (fiber) composite, the resin is cured (crosslinked) through heating, typically carried out in an autoclave or a press. During the autoclave curing process the temperature and the pressure are controlled in order to achieve the desired degree of cure of the resin and proper compaction of the part. Conventional controllers use a pre-specified temperature cure cycle. The cure cycle may be given by the material manufacturer, obtained from empirical methods, or from

analytical models. Self-directed control can be obtained with the use of expert systems, which replaces the pre-programmed cycle and performs on-line control of the process based on changes in process and material states. Such self-directed control can accommodate variability in material and processing since the temperature set point of the autoclave is determined on-line. Current expert system control decisions permit only heating, cooling or holding the autoclave temperature [43]. It is suggested that an artificial neural network may capture the decision-making strategy and provide a continuos output of temperature set point adjustment. Although artificial neural networks have been successfully used for modeling non-linear systems [2, 10, 15, 16, 17, 25, 26, 28, 29, 30, 33], it has not been used for such control purpose.

This work studies the feasibility of using a back propagation neural network to replace the expert system decision making mechanism. The network is trained by an operator's decisions on temperature set point adjustments under various process states such as temperature rates, gradients, etc. The trained network is then used to control a simulated autoclave process.

CHAPTER II

LITERATURE REVIEW

In the search for better ways to control the autoclave curing process, various methods were proposed by different researchers. This chapter summarizes some of these works. Although the application of neural networks to solve industrial problems is recent, there is a great amount of work published in the open literature. A brief review of artificial neural networks and their applications to solve chemical engineering problems is also presented.

Autoclave Curing of Composite Materials

The production of composite materials consists basically of arranging the uncured resin-fiber mixture (prepreg) into the desired shape and then curing the material by applying heat and consolidating it to its final shape by applying pressure. Heating is required to initiate and maintain the curing (crosslinking) reaction which hardens the resin. Pressure application not only removes excess resin and consolidates the part, it also prevents void formation due to volatile components in the resin [1].

For producing a flat composite panel in an autoclave, layers of prepreg are packed together with other materials and sealed in a vacuum bag as shown in Figure 2.1. The

bleeder absorbs excess resin from the prepreg layers during the cure. The breather allows uniform application of vacuum over the lay-up and removal of entrapped air or volatiles produced during the cure. The vacuum bag allows application of vacuum in the bag and also autoclave pressure to the lay-up. The dam is used to prevent resin flow from the edges [24]. The lay-up is then placed in an autoclave for the curing and consolidation process.

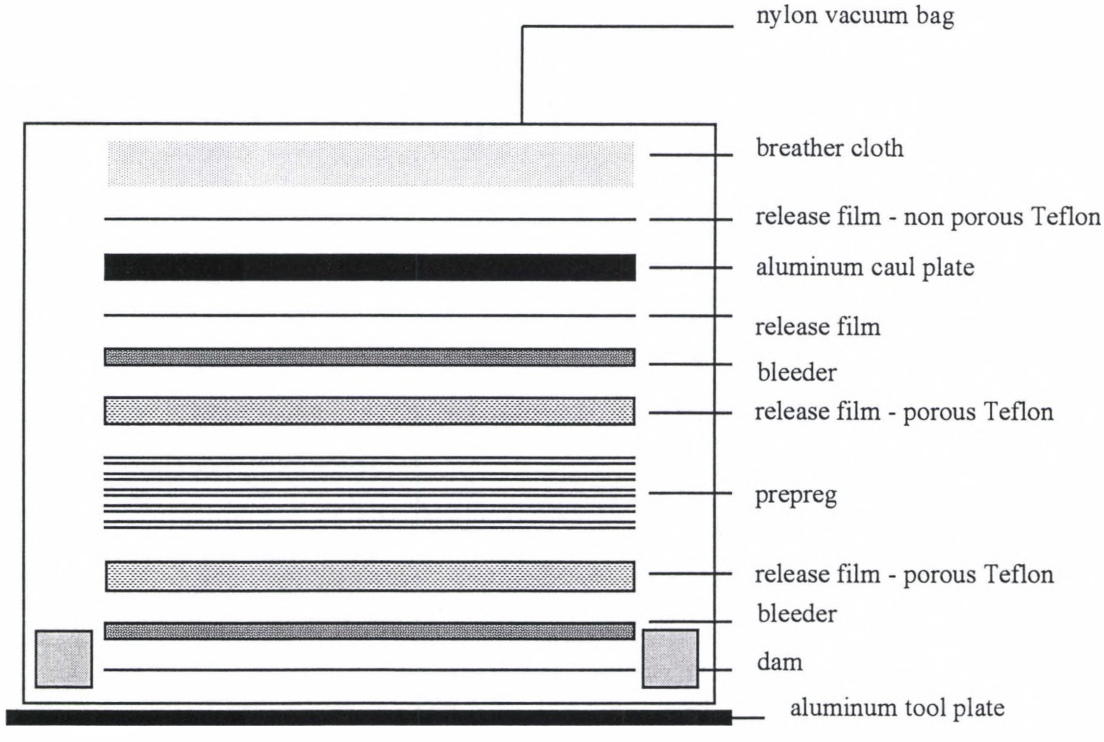


Figure 2.1. Schematic of the Prepreg Lay-up

Typically, the only controls one has are the autoclave temperature, autoclave pressure, and the bag vacuum. This work concerns the temperature control only. When the temperature is not controlled properly the reaction may not be complete or thermal runaway may occur [14]. In a conventional controller a pre-specified temperature cure cycle is used. There are several ways to select the temperature cure cycle for a given composite material [1]:

1. Cure cycles are recommended by the material manufacturer. Although it is an easy method to implement, the cycles may not be appropriate for most industrial applications since they are determined based on small samples.
2. Empirical methods are used to determine the optimal cure cycle. This trial-and-error method is not very efficient and leads to material losses and waste of operator and equipment time.
3. Analytical models of the process can be developed and used to simulate the process behavior which allows determination of an adequate cure cycle by trial-and-error on this simulated process. An optimal solution is difficult to find due to simplifications made in the model development.

The use of a pre-specified cure cycle based on time does not allow for variations in raw material properties or variations in processing conditions. To overcome these limitations self-directed process control is used. Self-directed control requires monitoring the cure process constantly and manipulating process conditions to obtain the desired cure state. The cure state is defined by the degree of cure (0 to 100%), a critical parameter in

the process control [39], which influences the glass transition temperature and hence the mechanical properties of the composite.

Whether one uses conventional controller or a self-directed controller, sensors are required to monitor the process state. Thermocouples are typically used for temperature measurement [14, 33]. Dielectric sensors [15] are used for resin viscosity, from which the degree of cure is inferred. Fiber optic sensors not only allow determination of resin viscosity but also temperature, pressure, strain and voids [12, 13]. Thermopile sensors have been used for measurements of heat release rate [11]. There are also sensors for thickness and resin pressure measurements [35].

A conventional controller, as shown in Figure 2.2 (a), compares sensor input with pre-programmed temperature cure cycle and makes set-point adjustments to track the pre-programmed cycle. A self-directed controller replaces the pre-programmed cycle by rules of controlling the process as shown in Figure 2.2 (b). Examples of such self-directed systems are given by Lee et al. [43], Saliba et al. [42], Ciriscioli and Springer [1], Wu and Joseph [8], and Buczek [35]. In general , these systems obtain process state through the use of sensors and decide on control actions (set point adjustment) based on programmed rules. The rules are typically programmed under an expert system shell. The systems developed by Saliba et al. [42] and Wu and Joseph [8] also incorporate mathematical models of the process. This study is an attempt to replace the rule-based controller by an artificial neural network as shown in Figure 2.2 (c).

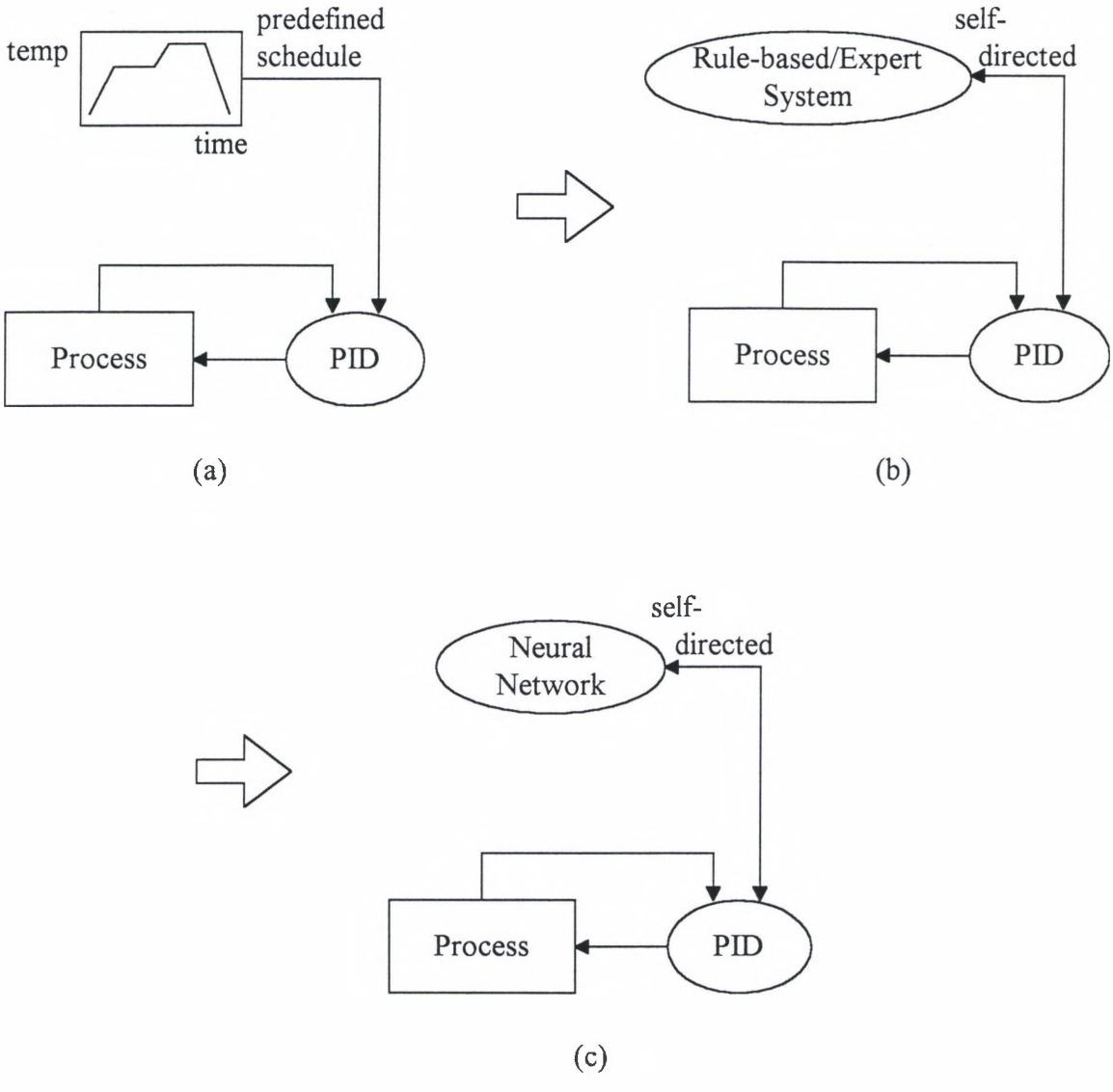


Figure 2.2. Conventional (a), Rule-Based/Expert System (b), and Neural Network Controller (c)

Artificial Neural Networks

The development of artificial neural networks started with an interest in simulating the human brain. The main neural network paradigms available for practical engineering

applications consist basically of a method for nonlinear mapping between a set of input values and a set of desired output values [20]. Some methods can be used for classification problems where the input vector can be represented by binary values (i.e., 0 or 1's). Other methods can be used for mapping continuous input vectors. The initial step of the development of neural networks is the training phase. The training phase consists of repeatedly presenting a set of input data and the desired results to the network and adjusting internal parameters (weights) to minimize the error between the desired output and the network output. During this training phase the neural network learns the function behavior [16]. Neural network paradigms may require supervised or unsupervised learning. In supervised learning the input vector and the desired output vector are presented during the training procedure. For unsupervised learning the input vectors are presented and the neural network groups them in a way that similar inputs will result in similar results.

The training algorithm can be recursive or non-recursive. The non-recursive, or feed-forward algorithm is easier to implement into a computer program and gives good approximations for static mappings. The recursive, or feedback algorithm uses previous information about the system which provides an interesting tool for simulating the dynamic behavior of systems [10].

Most neural network models present a neuron as a basic processing unit. The mathematical model of a neural network is built from a neuron-like structure. The neuron (node) has input connections that are processed by an activity level (activation function) to produce an output value. Each connection has an associated weight that can be excitatory

(greater than zero) or inhibitory (smaller than zero). A simple neuron-like structure is shown in Figure 2.3.

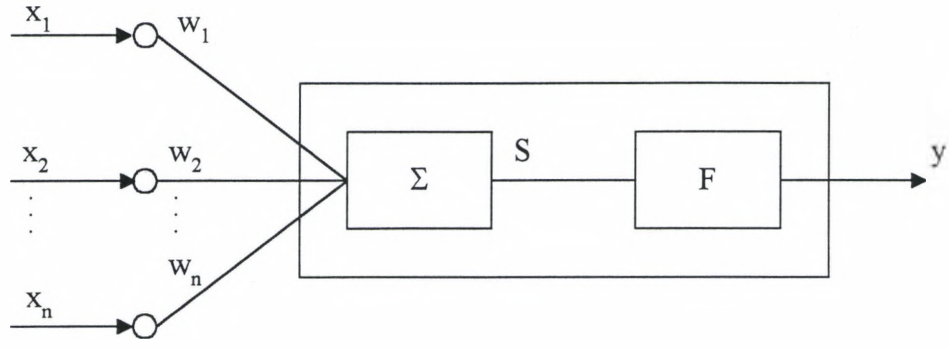


Figure 2.3. A Neuron-Like Structure

Referring to Figure 2.3, x_i is the input vector; w_i is the weight vector that represents the strength of the connection; S is the sum of the weighted inputs to that neuron; and y is the neuron output calculated by using the activation function F. Mathematically these can be expressed as:

$$S = w_1x_1 + w_2x_2 + \ldots + w_nx_n = \sum_{i=1}^{n} w_ix_i \tag{2.1}$$

$$y = F(S) = \frac{1}{(1+e^{-S})} \tag{2.2}$$

The activation function shown in Equation 2.2 is the most used function. It is a nonlinear function that compresses the range of S so that the output, y, lies between zero and one

[21] as shown in Figure 2.4.

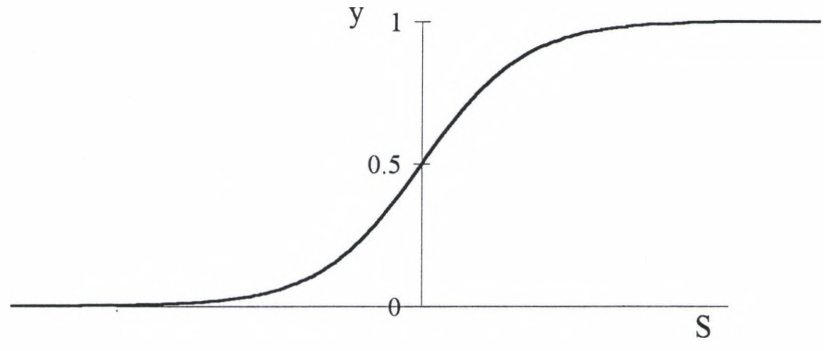


Figure 2.4. Sigmoidal Activation Function

Several paradigms were developed using different combinations of the neuron-like structure, alternative activation functions, and other modifications by various authors [4, 21, 23, 25]. Only the back propagation neural network will be discussed as it is used for this study.

The back propagation algorithm is by far the most widely used network for practical applications. Approximately 90% of the reported applications of neural networks use back propagation algorithm. Its simple and straightforward technique for minimizing the error is the main reason for its success.

Backpropagation Neural Network Learning Algorithm

Backpropagation neural network, proposed by Rumelhart et al. [3], is constructed from the basic neuron structure shown in Figure 2.3. Layers of neurons are connected as shown in Figure 2.5. The input layer does not perform any calculation (broadcaster layer).

The input neurons are connected by links to the neurons in the hidden layer. More than one hidden layer may exist. Neurons from the last hidden layer are connected to neurons in the output layer. More than one neuron may exist in the output layer. The output of a network (output layer) is a function of the weights of the links and of the network inputs. Bias neurons are added to the input layer and hidden layers. The input to a bias neuron is always one.

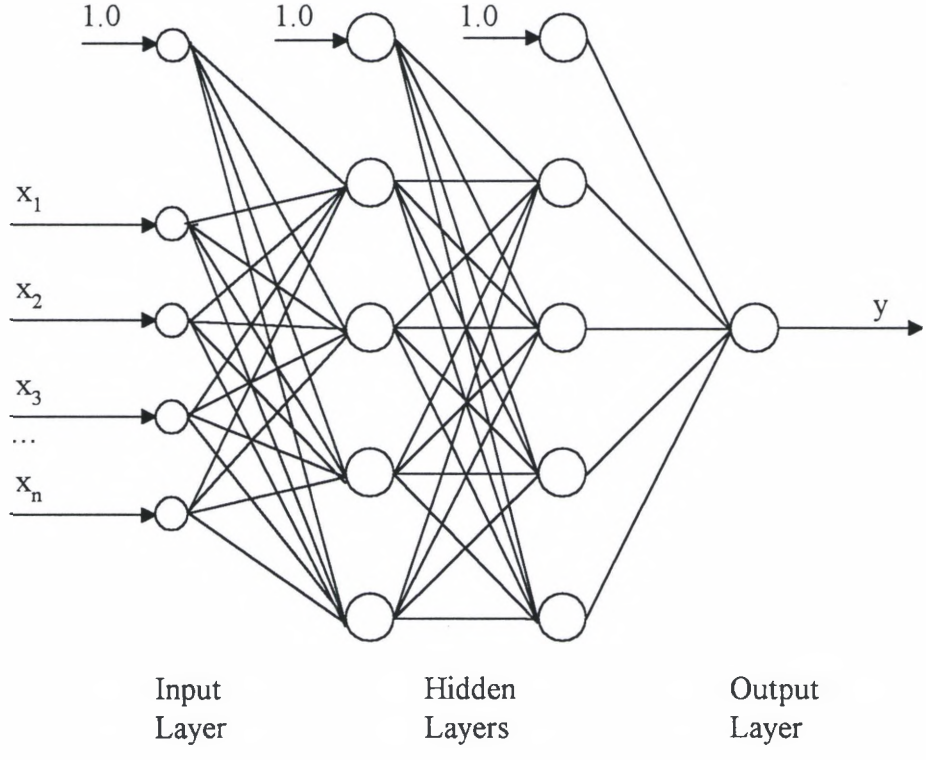


Figure 2.5. Back Propagation Neural Network Architecture

During the training phase the weights are adjusted in order to produce the desired outputs. A training vector consists of a set of inputs (x vector) and desired outputs (y vector), the paired input-output vector is called a training pair. A large number of

training pairs should be available to train the network vectors. The group of training pairs used in the training is called a training set. The main steps of the training are [21]:

1. Select training set.
2. Select initial values of the link weights (random small values).
3. Calculate y from Equations 2.1 and 2.2.
4. Calculate the error between the network output and the desired output.
5. Adjust the link weights of the network in a way that minimizes the error.
6. Repeat steps 3-5 until error reaches an acceptable value.
7. When the error reaches an acceptable value the network is said to be trained. The network performance should then be tested by using a testing set different from the training set.

There are several ways to perform step 5. A brief description of the gradient descent technique, which is the error minimization method used for this study, will be given. In the gradient descent technique the weights are updated to minimize the error function (J),

$$J(w) = \sum_{p} E_p = \sum_{p} \sqrt{(y_c - y_d)^2} \tag{2.5}$$

In Equation 2.5, the subscript p represents patterns in the training set; y_c is the calculated output; and y_d is the desired output. Until the desired error is obtained or the maximum number of iterations (epochs) is achieved, weights are updated according to

$$\Delta_p w_{ji} = \eta \delta_{pj} + \alpha \Delta_{p-1} w_{ji} \tag{2.6}$$

where $\Delta_p w_{ji}$ is the change in the weight from node i to node j at pattern p; η is a gain term also called the learning rate; α is a momentum term; the error signal δ_{pj} is a measure of the distance from the activation level of node j to its desired level.

The learning rate regulates the rate at which weights are changed during training. If the learning rate is set to 0.5 then the weight change is a function of half of the error. Larger values of learning rate produce larger weight changes and faster training. Momentum is used to determine the proportion of the last weight change into the new weight change. It is used to avoid oscillation of weight changes. Momentum provides a smoothing effect and permits the use of larger training rates.

The error signal, δ_{pj}, for an output node is calculated following the equation

$$\delta_{pj} = (d_{pj} - y_{pj}) f_j'(a_{pj}) \tag{2.7}$$

where d_{pj} is the desired activation level of node j for input pattern p and $f_j'(a_{pj})$ is the derivative of the output function for node j with respect to activity generated by the input pattern p. And, for a neuron in the hidden layer,

$$\delta_{pj} = f_j'(a_{pj}) \sum_k \delta_{pk} w_{kj} \tag{2.8}$$

where the summation is over all k nodes to which node j sends output.

After equations 2.7 and 2.8 are differentiated the error signal can be calculated; for the hidden layer(s)

$$\delta_{pj} = y_{pj}(1 - y_{pj}) \sum_k \delta_{pk} w_{kj} \tag{2.9}$$

and for the output nodes

$$\delta_{pj} = (d_{pj} - y_{pj})y_{pj}(1 - y_{pj}) \tag{2.10}$$

The procedure consists of first propagating the inputs forward through the network to calculate the network output. The calculated output is compared to the desired output to produce an error signal. The error is back propagated through the network to update the weights. This procedure can be used to represent several different non-linear function behaviors.

Applications of Neural Networks in Chemical Engineering Processes

The ability to represent non-linear function behavior makes neural network a powerful numerical modeling tool in intelligent process control [16] and in process-fault diagnosis. It can also be used in applications such as fault detection, process control, process design and process simulation. Some examples of its application in chemical engineering are described in this section.

Hoskins and Himmelblau [10] indicate that artificial neural networks are particularly suited for chemical engineering tasks requiring pattern recognition or continuous input-output control in processes with uncertain models and data. They have developed a neural network for diagnosis and fault detection for a system of three continuous stirred tank reactors in series. Good results were obtained within the range studied.

A process model of the autoclave curing of composite materials is described by Joseph and Hanratty [2]. A back propagation neural network was used to represent the

batch manufacturing process. The system has been tested on-line. Their network receives several inputs from the process and predicts the laminate thickness and the maximum void size in the laminate.

An isothermal CSTR reaction neural network model was developed by Bhat et al. [26]. They compared the neural network model prediction to previous model results. The neural network input were the scaled feed composition and reactor space time. The outputs were the dimensionless product concentrations. Historical values were used to train the network. Good accuracy was obtained with the network model.

Neural networks are used in process control when the system to be controlled is non-linear and standard techniques do not give satisfactory results. A classification of the applications of neural network to process control was presented by White and Sofge [19].

The system proposed in this work consists of a supervised control system where the process is monitored constantly and control actions are taken according to the process state.

CHAPTER III

NEURAL NETWORK CONTROLLER AND AUTOCLAVE PROCESS SIMULATOR

The controller developed in this study is a neural network that controls the temperature of a simulated autoclave. A neural network controller (NNC) consisting of a general purpose back propagation neural network (BPNN) and data input/output (I/O) support has been developed. Although this NNC can be used in real-time for monitoring and control with appropriate I/O modifications, the current version uses a simulation program to simulate the autoclave and composite temperature responses. These tools, the NNC and the simulator, are described in this chapter along with consideration of process variables to be monitored.

Neural Network Controller

A general purpose BPNN capable of handling multiple input/output neurons and multiple hidden layers forms the core of the NNC. The I/O support includes file I/O for training data input and file storage of trained link weights during the training phase, and serial port I/O for simulated sensor input and control set-point output when the NNC is operating in the controller mode. The complete source code listing of this program is included in Appendix A.

The BPNN was developed using a neural network tool, NeuroWindows™ (NW), from Ward Systems Group, Inc. NW is a Dynamic Link Library (DLL) for the Windows environment. It is a collection of neural network functions that facilitates the construction and training of various types of neural networks. For I/O support under the Windows™ environment, Visual Basic™ (VB) from Microsoft® Corporation was used as the programming language. VB also facilitates the Graphical User Interface (GUI) under the Windows™ environment.

A brief description of the NNC program is given here. Program procedures that are contained in NW are expressed in upper-case and smallcaps letters (e.g. SMALLCAPS) and those developed under this study are in mixed upper- and lower-case letters (e.g. UpperLowerCase). The subroutines associated with the BPNN are CreateNet, TrainNet and RunNet. Subroutine CreateNet constructs the BPNN using the NW functions MAKENET, MAKESLAB, and MAKELINK. Each slab represents a layer of neurons. MAKELINK establishes the links between the slabs. NW functions PUTSLAB, BPPROPAGATE, BPEVALUATE and BPTRAIN are used in subroutine TrainNet. The training set, which contains several hundred training pairs (input vector and desired output vector), is first read in by subroutine ReadTrainingSet. Each input vector is fed to the network by PUTSLAB. The input vector then propagates through the slabs by BPPROPAGATE. The output vector from the last slab is compared with the desired output vector by BPEVALUATE. Then the network is trained by back propagation (BPTRAIN) to minimize the errors. These steps are performed on each training pair until the entire training set is

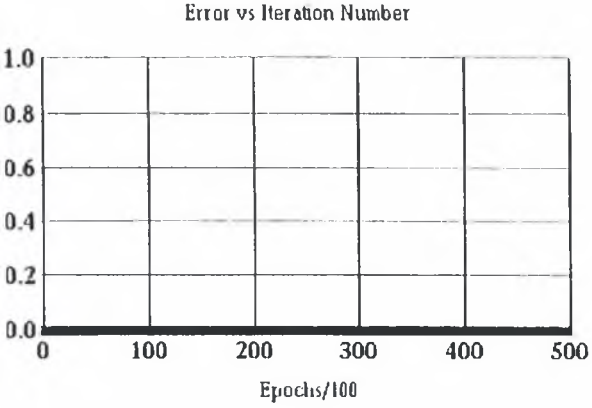
processed. Such an iteration is also called an epoch. Many epochs are required to train the network. Subroutine RunNet is used to predict output after the network has been trained. RunNet uses NW functions PUTSLAB to pass input to the network, BPPROPAGATE to propagate through the slabs, and GETSLAB to pass back the output vector.

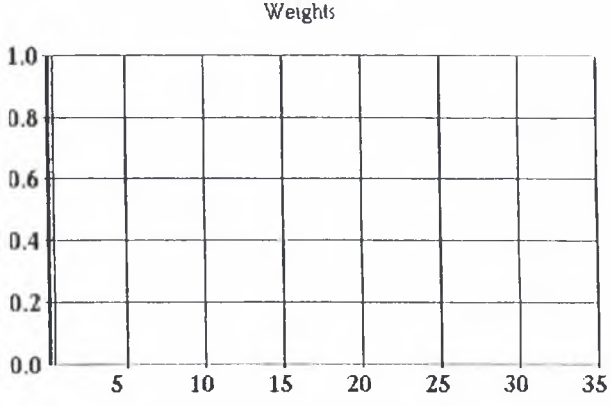
The NNC program presents to the user a graphical screen (a VB form) as shown in Figure 3.1. On this form, the user enters the total number of epochs (Epochs) to train the network and the number of neurons in the hidden layers of the network (Hidden). For multiple hidden layers, the number of neurons in each layer is separated by a comma. The number of input and output neurons (Inputs and Outputs) are set by the training data file (the TRN file) and can not be changed at run-time. Their values are echoed upon loading training data by selecting Train on the form. A dialog box appears for the user to enter a training data file name. The training data file structure is commented in subroutine ReadTrainingSet. The training data files used for this study are included in Appendix B.

The NNC program allows randomization of the sequence of the training pairs after each epoch by selecting Randomize on the form. The link weights are stored to a file (the LNK file) at an interval specified by the user. This parameter also determines how often the chart display of the network error is updated. Chart displays, for record-keeping purposes, can be printed by double-clicking on the graph.

Selecting Load Links allows the user to retrieve previous trained network link weights in an LNK file. In this case, the value of Epochs on the form specifies which set of link weights in the LNK file is to be loaded. After either Train or Load Links the network

Figure 3.1. NNC Graphical User Interface


Error vs Iteration Number
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Epochs/100


Weights
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
30
35

can be exercised by selecting Run - File or Run - Sim. Run - File reads input data from a file, runs the network, and writes the output data to another file. It serves as a general-purpose neural network. For this study it is mainly used to test the network programming by using data generated by a known function. Run - Sim reads input data from a serial port, runs the network, and sends the output data to the serial port. Run - File and Run - Sim differ in I/O support and that Run - Sim is specialized for the simulator.

Autoclave Process Simulator

The autoclave process simulator consists of a computer program developed by Lee [18], which is based on the "CURE" program developed by Loos and Springer [38]. The thermochemical model, which describes the part behavior, solves the one dimensional heat transfer equation. Heat source (exothermic reaction) and convective boundary conditions are considered. The model is represented by the partial differential equation:

$$\rho C_p \frac{\partial T}{\partial t} = -\frac{\partial}{\partial x}\left(-k\frac{\partial T}{\partial x}\right) + \rho r w \Delta H \qquad (3.1)$$

where ρ is the composite density (kg m^{-3}); C_p is the composite heat capacity (J kg^{-1} K^{-1}); T is the local temperature (K); t is time (s); x is position (m); k is the composite thermal conductivity (J m^{-1} K^{-1} s^{-1}); r is the reaction rate (s^{-1}); w is the resin mass fraction; and ΔH is the resin heat of reaction (J kg^{-1}). Equation 3.1 describes the heat transfer in the part during the curing reaction. The convective boundary condition for autoclave curing is

$$-k\left.\frac{dT}{dx}\right|_{\text{at surface}} = h\left(T - T_a\right) \tag{3.2}$$

where h is the heat transfer coefficient (J m^{-2} K^{-1} s^{-1}) and T_a is the autoclave temperature (K). The simulator was used for generating the training set and for testing the neural network controller. When used for generating training set, the simulator accepts autoclave temperature set-point adjustment from the keyboard and reports laminate temperature at user specified nodes in graphical display. When used with the neural network controller, the data I/O is through the serial port.

Process Monitoring

During the curing process the temperature in the laminate and inside the autoclave were monitored. In addition to monitoring the autoclave temperature (Taut), temperatures at four positions within the laminate were also monitored as shown in Figure 3.2. The laminate sensors were located at laminate centerline (Tmid), laminate top (Ttop), and two locations on either side of the centerline at one node distance away from the midpoint (Tm+, Tm-). Data provided by these five sensors were used to obtain the derived data used for control purpose.

These five sensor inputs were chosen because pertinent process states described by the energy balances in Equations 3.1 and 3.2 can be derived from them. The part temperature, which is the controlled variable, is represented by Tmid. The temperature difference between the laminate and the autoclave (Tmid – Taut) provides information on heat transfer by convection across the boundary (Equation 3.2). The temperature

difference between the laminate top and midpoint (Ttop – Tmid) represents the temperature gradient in the part, which is the driving force for heat transfer by conduction in the part. The temperature rate (dT/dt) represents the dynamics of the system. The last

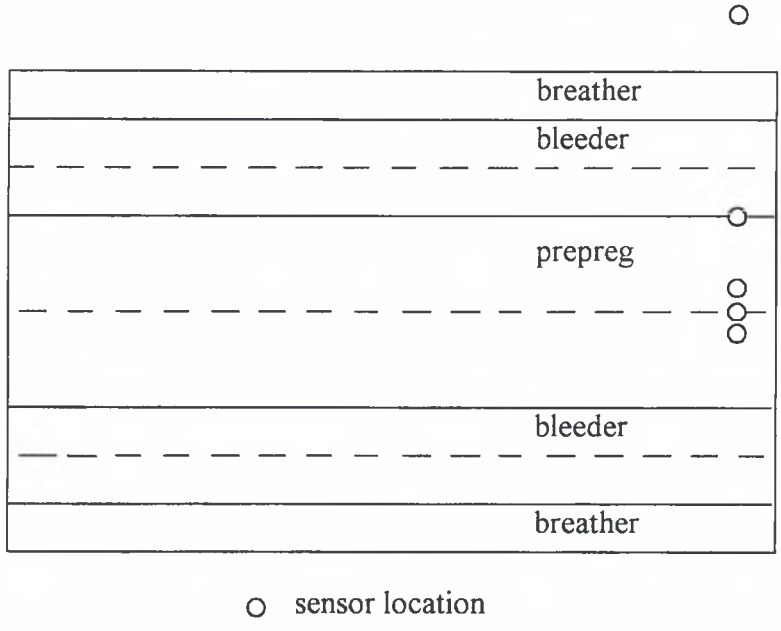


Figure 3.2 Lay-Up and Temperature Sensor Locations

term in Equation 3.1 represents the heat release rate due to the exothermic reaction. The rate is in general dependent on both temperature and the degree of cure of the resin. A quantity proportional to the heat release rate can be obtained from dT/dt and (Tm+ -2 Tmid + Tm-) according to method described by Lee and Rice [11]. These five derived data, summarized in Table 3.1, were used as the input vector to the neural network controller.

Table 3.1

State Variables

Variable	Description	Analogy in the model
Tmid	Actual laminate temperature	Heat transfer by conduction in the laminate
Tmid-Taut	Temperature gradient	Heat transfer by convection in the autoclave
Tmid-Ttop	Laminate temperature gradient	Heat transfer by conduction in the laminate and insulation
dT/dt	Temperature rate	Dynamics of the system
Hr	Heat release rate	Reaction exotherm, source of heat, function of the degree of cure (history)

In summary, the curing process was simulated by a one-dimensional heat transfer model that included heat source and convective boundary conditions. The neural network controller derived the process state from temperature data at five locations to arrive at a control decision in terms of autoclave set-point adjustment. The set-point adjustment was then sent to the simulated autoclave, which set the autoclave temperature for the next time step of the simulator program execution. Figure 3.3 illustrates this closed-loop self-directed control system.

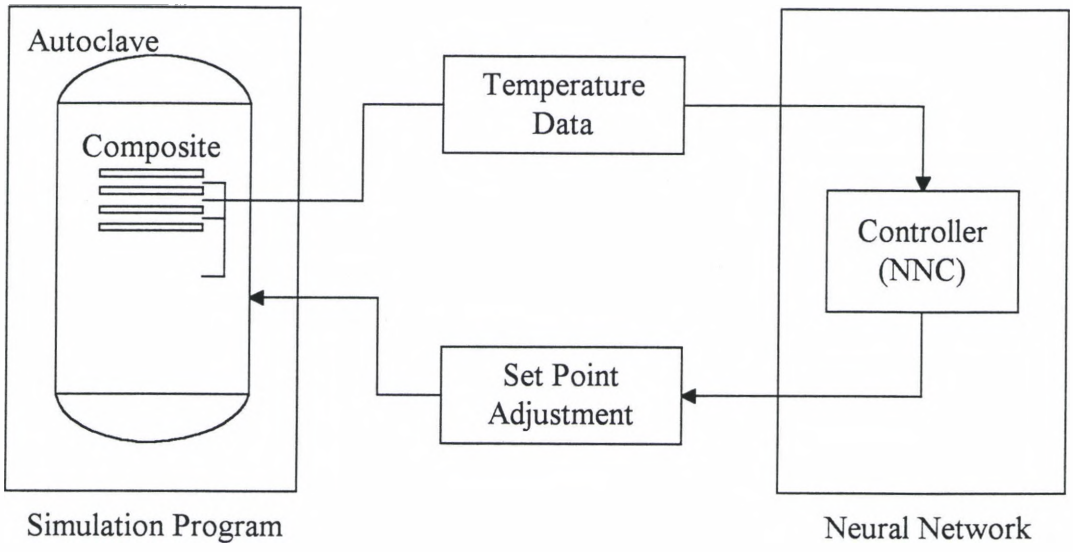


Figure 3.3. Closed-Loop Self-Directed Neural Network Control System

CHAPTER IV

TRAINING AND TESTING OF THE NEURAL NETWORK CONTROLLER

The neural network controller performance is presented in this chapter. First the neural network was trained and then used to control simulated curing of parts of different thicknesses. The sequence used is presented and the results obtained in each step are discussed.

Neural Network Controller Training

The neural network controller (NNC) was developed using the sequence shown in Figure 4.1. The supervised back propagation neural network (BPNN) learning paradigm was used to perform the network training in this study.

The training set was created using the simulator program and the operator's knowledge. The operator controlled the simulated autoclave temperature to obtain an acceptable cure cycle. This cure cycle, represented by autoclave set point adjustments, constitutes the desired output of the neural network. Five process variables as described in Chapter III form the input vector to the neural network. These input and desired output values were stored at 1-minute intervals during the simulation run. Upon completion of a successful simulation, a training set was created.

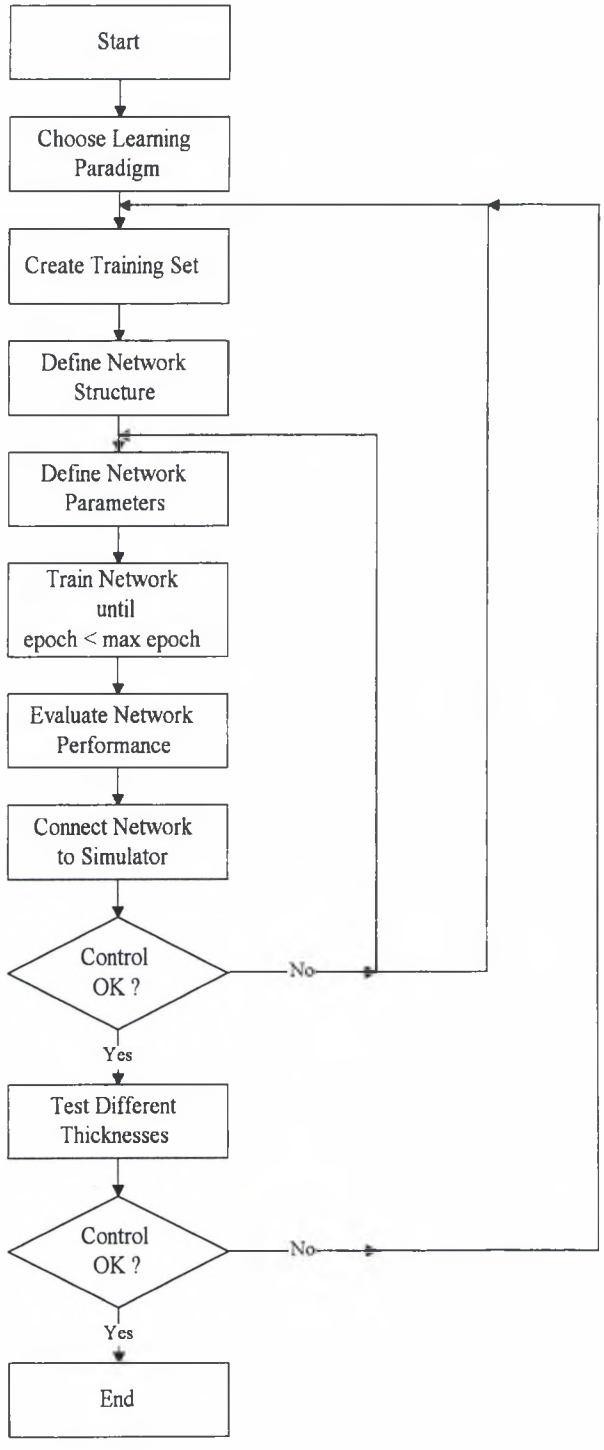


Figure 4.1. Neural Network Controller Development Procedure

The control objectives were to heat the part as fast as possible to its cure temperature of 350°F; to avoid excessive temperature rise (not to exceed 500°F) in the part due to reaction exotherm; and to have the part at the cure temperature (or higher) for a long enough period of time to ensure complete cure. There were also constraints on the autoclave temperature range of 80 to 450°F, heat-up rate limit of 10°F/min, and cool-down rate limit of −10°F/min. The operator's decisions concerning the set point adjustments relied on the autoclave/laminate temperature differences, laminate temperature gradients, and heat released by the reaction. All necessary variables were shown on the computer screen, along with strip chart-like graphical display of temperatures, during the simulation run. Although the numerical display provided useful information, it was the operator's intuitive interpretation of the graphical display that lead to the decision making. The operator's intuition and experience had an important role in the decision making. When monitoring the present and past process state (process dynamics) displayed on the computer screen, future conditions were anticipated and control action decisions made before an unacceptable condition could take place. It is this decision making process, which is difficult to quantify, that was attempted to be captured by the neural network.

After an acceptable training set had been created, the next step was to randomize the data. This was done to avoid memorization by the network. The learning algorithm may produce a set of weights that produce exactly the desired output when it has "memorized" the pattern but may not be able to produce a reasonable output when a

slightly different process state (noise) is presented. After the data had been mapped and randomized the neural network was then trained.

The training set defines the network architecture as far as the number of neurons in the first (input) and the last (output) layer of the network. The network at this point consisted of five input neurons and one output neuron corresponding to the training set created by the operator. At training time, one had the choice of the number of hidden layers and the number of neurons in each hidden layer. As the architecture of the network should be kept as simple as possible, the initial trials started with one hidden layer with five neurons. Additional neurons were added to the hidden layer in an attempt to improve the knowledge capture. Up to nine neurons in the hidden layer were used in the training. Also, a case with one additional hidden layer with five neurons in each layer was studied. A bias neuron was added to each layer. All neurons in each layer were connected to neurons in adjacent layers, resulting in a fully connected network. The learning rate was set to 0.7 and the momentum to 0.3 during the training. These values were kept constant for all cases studied.

A file containing training related information was created during the training session. The information saved consists of: links file name, training set file name, network architecture, parameter values, input data ranges, output data range, followed by weight values and root mean square (RMS) error recorded after every 100 epochs of training.

During training, the RMS average error was displayed in the graph form. All cases studied in this work were trained up to 10,000 epochs. Overtraining could occur when

too many iterations were performed. When this happened the network performance as a controller was not acceptable and link weights from a lower number of epochs were used. The weights variation can also be used for the analysis of the network performance.

The RMS average error was based on the desired output set point adjustment. In this study it is not a reliable measure of the network performance since the network is controlling a dynamic process. Rather, one should judge the network performance by the cure cycle developed by the controller. Once the trained BPNN is available it can be used in a straightforward manner to control the simulated autoclave curing process.

Neural Network Controller Testing

The autoclave (simulator) was heated to a maximum temperature of 450°F and cooled to a minimum of 80°F at rates determined by the neural network controller. The control started with the simulator sending the process state to the NNC. After processing the temperature information into the desired input data the NNC computed the set point adjustment and sent it back to the simulator. The same cycle was repeated for 400 steps in time, with each step representing 1 minute.

The NNC was first tested by processing a panel of the same thickness as that used for training. When this was successful, a validation was performed with a panel of different thickness. Otherwise, network link weights from different epoch numbers were tested. When no improvements were observed with all the intermediate trials, a new training set was created and the sequence shown in Figure 4.1 repeated.

Three different prepreg thicknesses were used in this study: 32, 128, and 256 plies. Training sets were created for 32- and 256-ply panels (Cases 1 and 2) and the intermediate 128-ply panel was used to test the NNC performance. The neural network training and testing are presented separately for each case in the following sections.

Case 1 Training

The sequence shown in Figure 4.1 was used to develop a NNC based on the simulated 32-ply panel. Input data for the simulator which include laminate thickness and physical properties of the prepreg are presented in Appendix C.

For a thin laminate the temperature gradient was very small. It can be seen from Figure 4.2 that the prepreg top and midpoint temperatures were very close. When developing the training set the heating rate was increased gradually from 1°F/min up to 9°F/min. When the autoclave temperature reached 250°F the heating rate was reduced gradually to 0°F/min. The midpoint temperature was held above 400°F to assure the cure is completed. The autoclave temperature was held constant for about 20 minutes before cooling started. The cooling rate varied form -1°F/min to -10°F/min, when the autoclave reached the minimum set point (80°F) the cooling rate was reduced to 0°F/min. The resulting training set file is presented in Appendix B.

A list of neural network inputs, the corresponding desired output and respective ranges for normalization is shown in Table 4.1. The input data was scaled to the range 0 to 1 and the output data to the range 0.1 to 0.9 as required by the back propagation

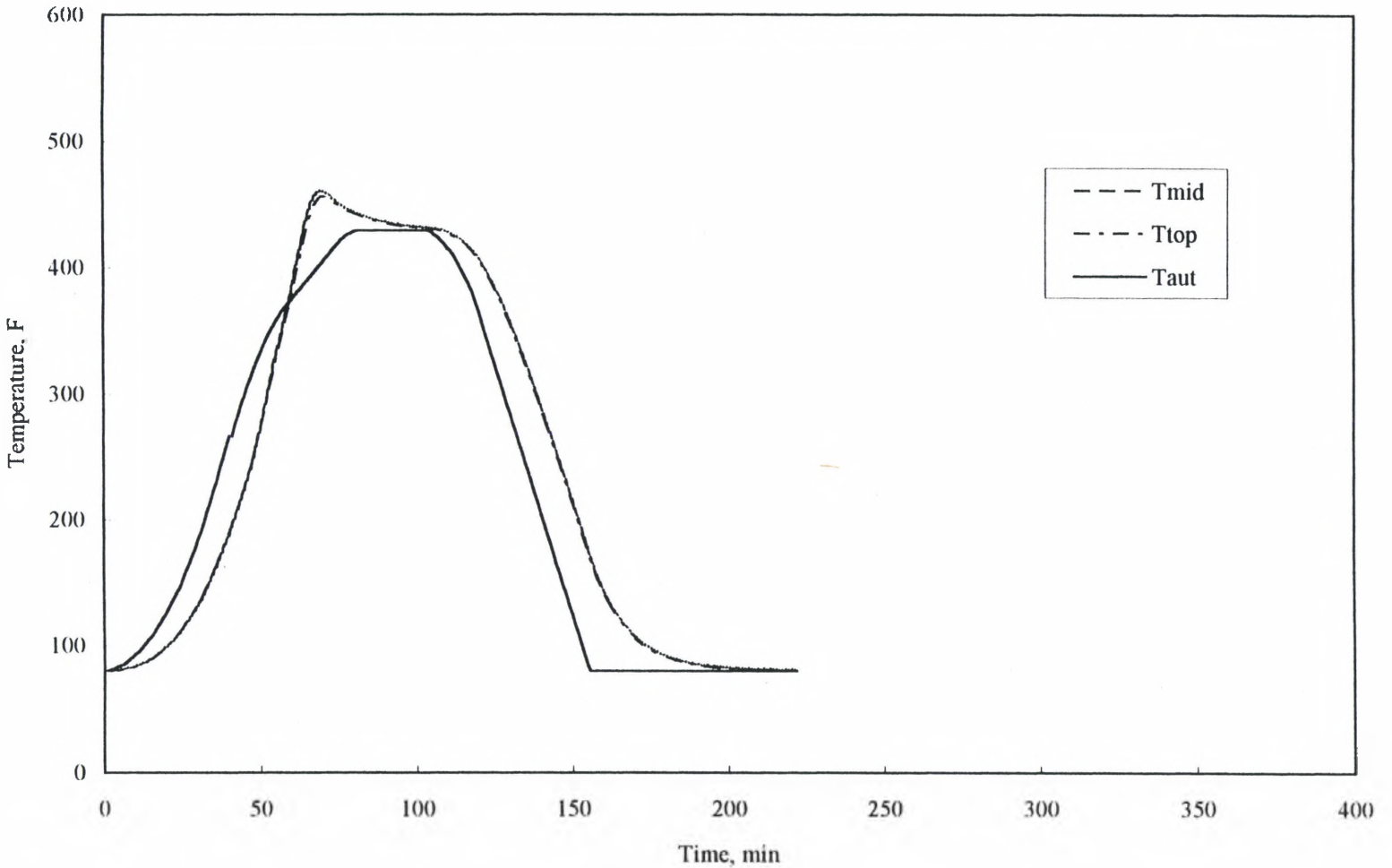


Figure 4.2. Case 1 Training Using Cure Cycle for 32-Ply Laminate

training algorithm. The initial ranges were chosen based on the maximum and minimum values from several simulation runs.

Table 4.1
Ranges Used to Scale the Input and Output Data

Variable	Initial Range	Mapped to:
Tmid	80 to 500°F	0 to 1
Tmid-Taut	−10 to 50°F	0 to 1
Tmid-Ttop	−100 to 200°F	0 to 1
dT/dt	−2.5 to 5°F/min	0 to 1
Hr	0 to 4°F/min	0 to 1
Tadj	−12 to 12°F/min	0.1 to 0.9

After creating the training set for the 32-ply cure cycle the network was trained with several architectures of different number of neurons in the hidden layer, and in one case, two hidden layers. The RMS error variations for these cases are shown in Figure 4.3 as a function of number of epochs trained. It can be seen from these curves that the errors initially decreased rapidly and then stabilized for all architectures studied. The simplest structure studied had five neurons in the hidden layer. The RMS error after 10,000 epochs for six hidden neurons was slightly lower than the case with five hidden neurons. However, further increase in the number of neurons did not show any improvement. In

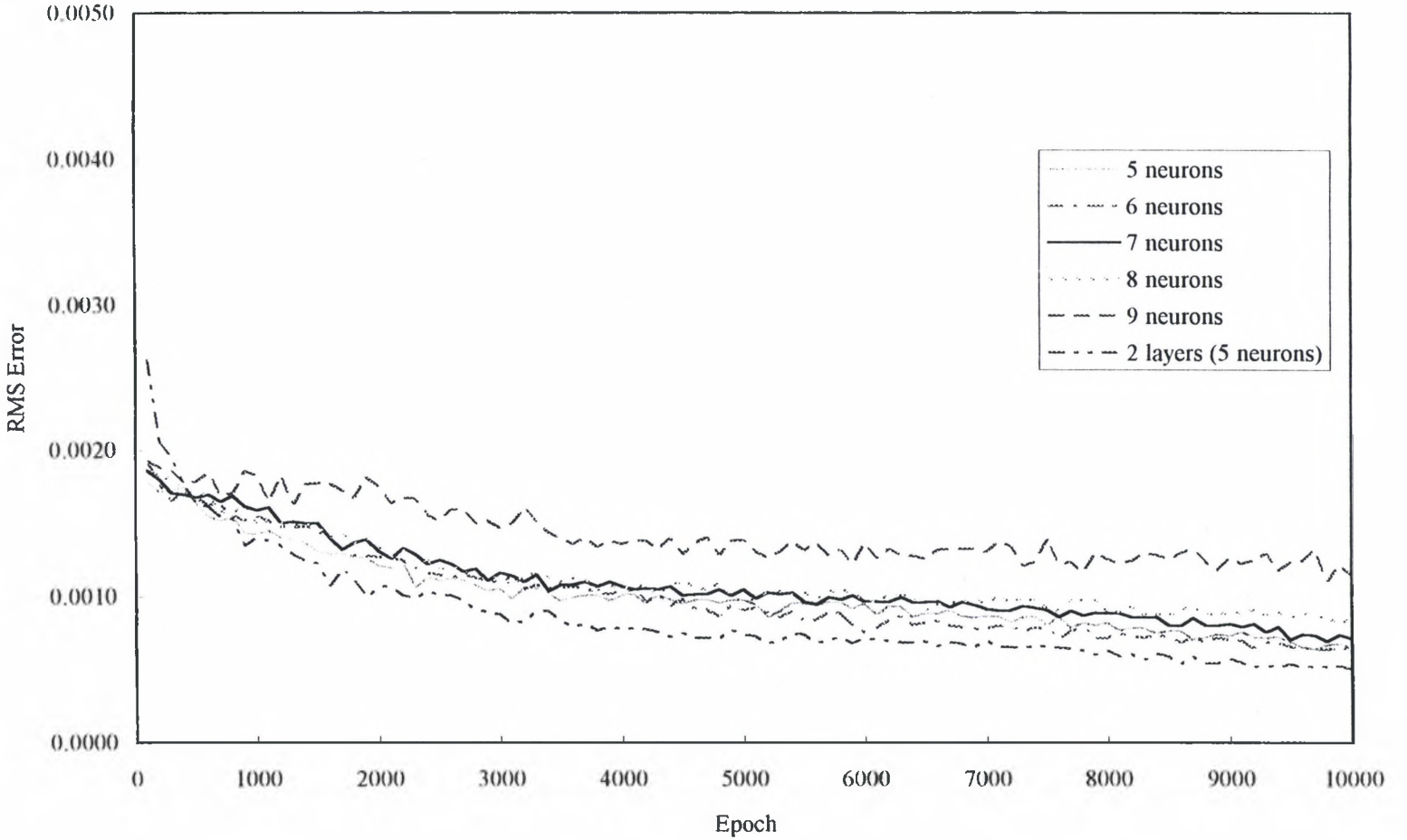


Figure 4.3. Case 1 RMS Error Variation, During Training

fact, the error was the highest with nine neurons in the hidden layer. The lowest error value was obtained when two hidden layers, with five neurons each, were used. Table 4.2 shows the number of links as a function of the number of hidden neurons. Increasing the

Table 4.2

Number of Links as a Function of the Number of Hidden Neurons

Number of Neurons				Total Number of Links	RMS Error After 10,000 Epochs
Input Layer	Hidden Layer 1	Hidden Layer 2	Output Layer		
5	5		1	36	0.000661
5	6		1	43	0.000651
5	7		1	50	0.000718
5	8		1	57	0.000864
5	9		1	64	0.001160
5	5	5	1	66	0.000509

number of neurons in the hidden layer increases the number of links and consequently the computational time required for training. The number of links created was 36 when five neurons were used in the hidden layer. The additional hidden layer increased the number of links to 66. Even though the two hidden-layer architecture resulted in the lowest RMS error, the increase in computational time did not justify the slight improvement in the RMS error. It was therefore decided that one hidden layer with six neurons would be used for further studies. This structure is shown in Figure 4.4. Weight variations during training for this chosen architecture are shown in Figure 4.5. From the graph it can be seen that the weight variations were in the range between −15 (inhibitory) and 11 (excitatory).

Although training of the neural network was carried out to 10,000 epochs and the RMS error diminished with increased number of epochs, testing of the NNC on the simulated autoclave showed that the best performance was obtained after 2,000 epochs.

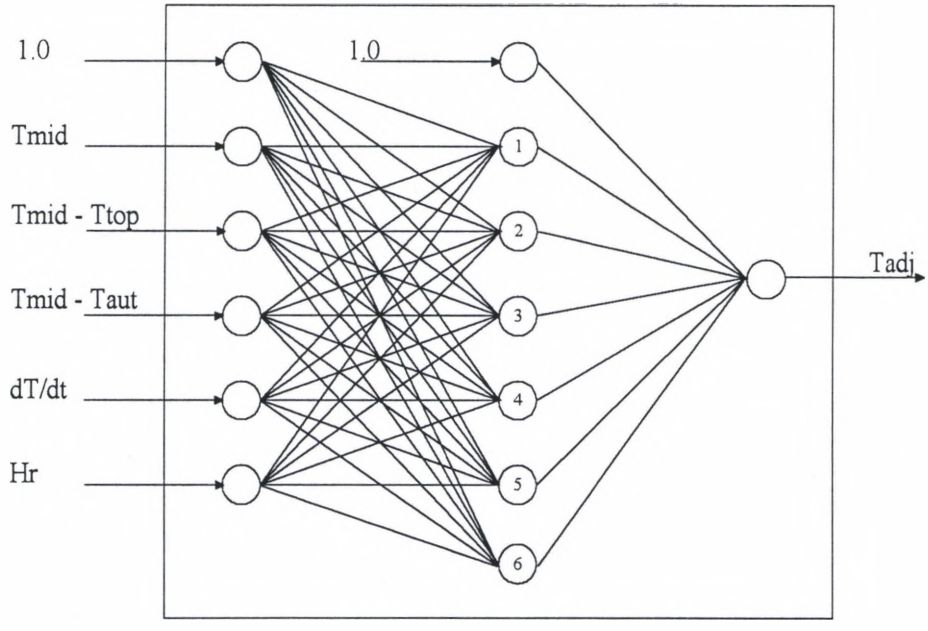


Figure 4.4. Back Propagation Neural Network Architecture Used

As was pointed out before, the performance of the NNC can not be judged by the RMS error. This lead to the decision to use the network link weights after 2,000 epochs of training for all subsequent NNC tests. These link weights for the 32-ply case are shown in Figure 4.6. The largest excitatory and inhibitory links are shown in Figure 4.7. It is interesting to see that while the last three inputs received much attention, the first two

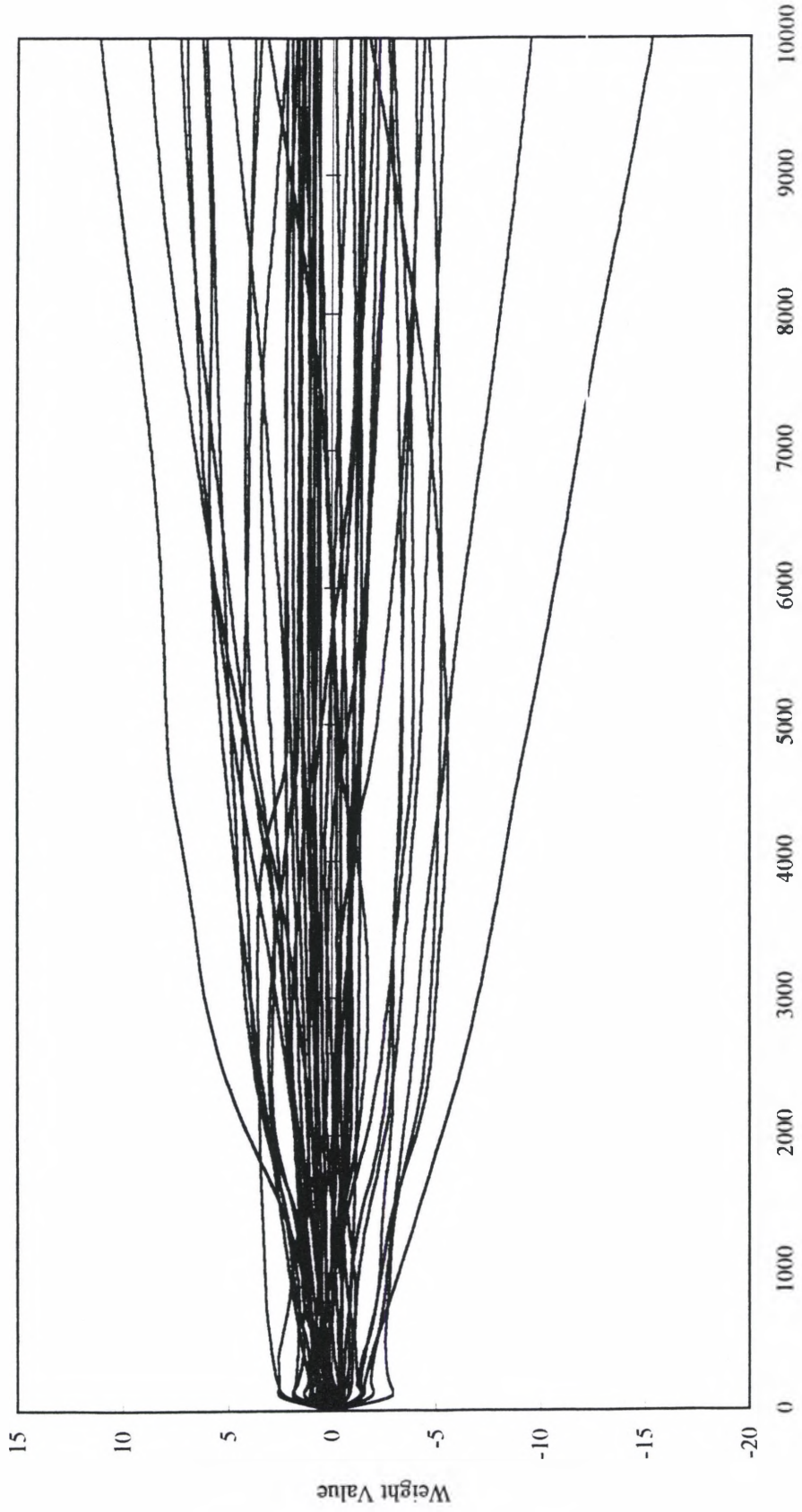


Figure 4.5. Weight Variations During Training of Case 1 NNC

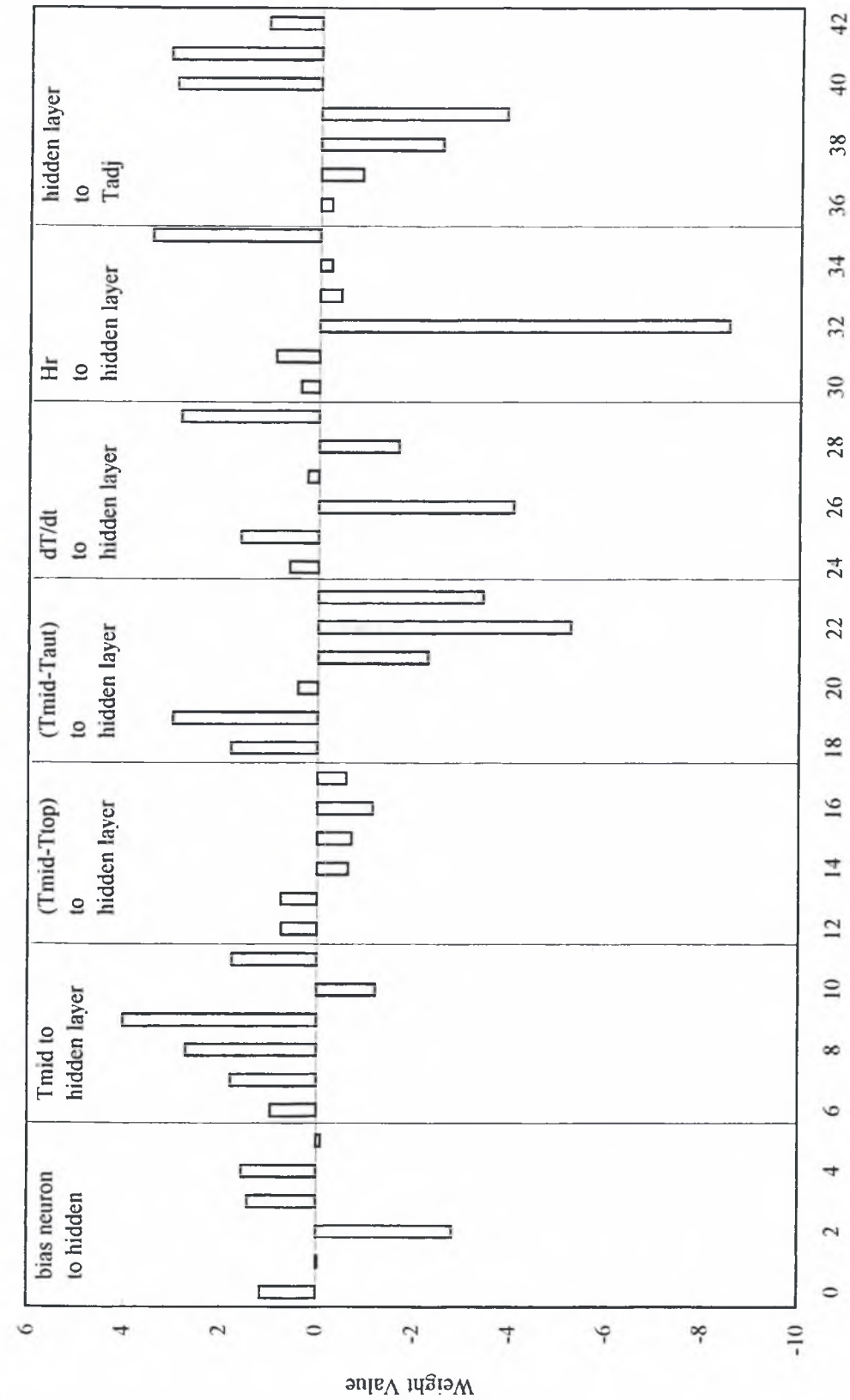


Figure 4.6. Weight Values for Case 1 NNC after 2,000 Epochs

especially (Tmid – Ttop), did not seem to matter much. It is thought that either the first two inputs were unimportant or the network had captured the operator's bias in decision making.

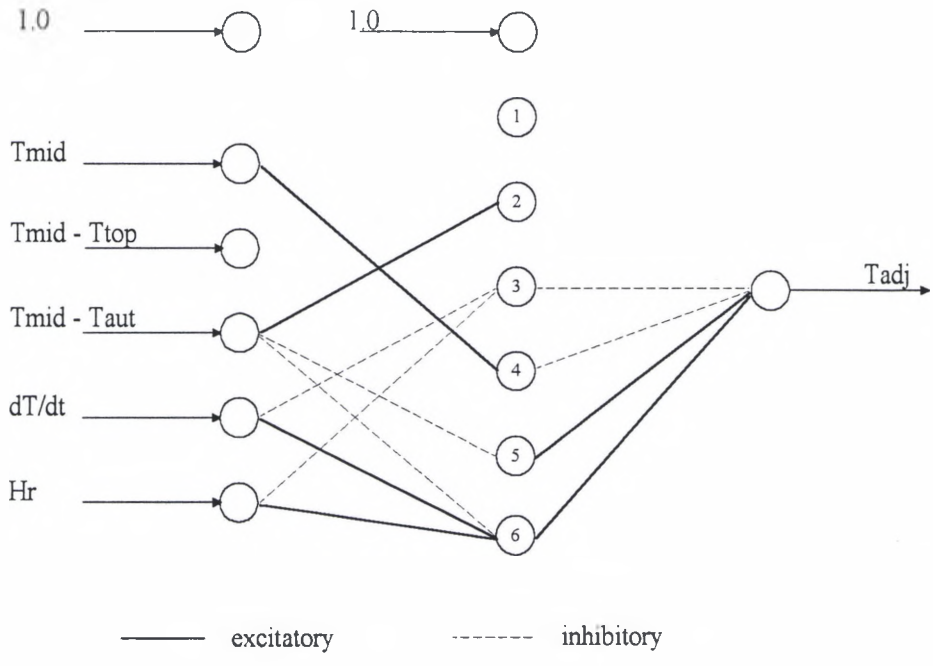


Figure 4.7 The Stronger Excitatory and Inhibitory Links of Case 1 NNC

Case 1 Testing

After the BPNN had been trained to control a 32-ply laminate its performance as a controller (Case 1 NNC) was first tested to see if it can reproduce the training cycle (Test 1a). For a 32-ply laminate the NNC was able to reproduce with considerable accuracy the training cycle (Figure 4.2) as can be seen in Figure 4.8. The NNC received the updated process state from the simulator every minute to make a control action

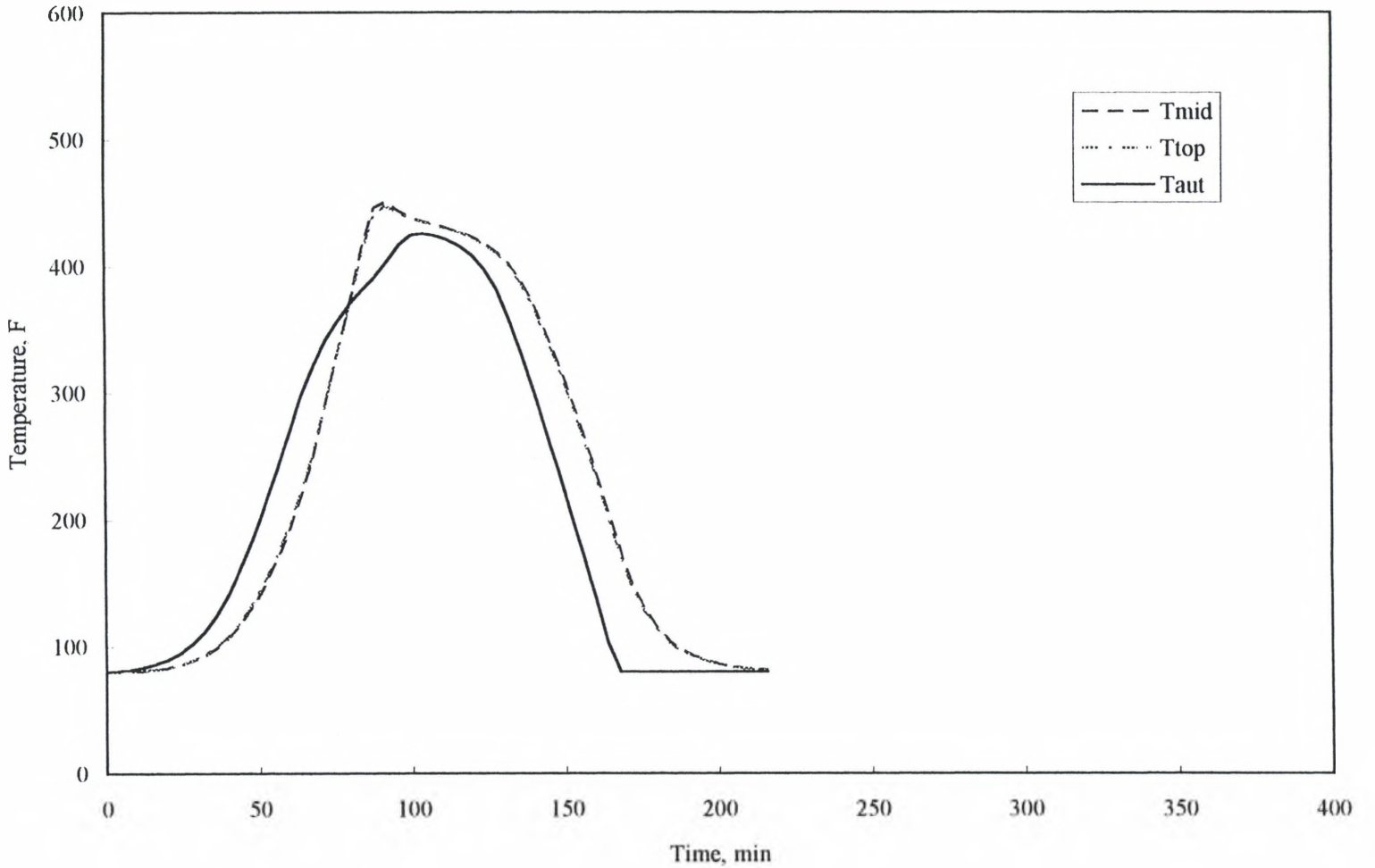


Figure 4.8. Test 1a, Case 1 Trained Network Controlling 32-Ply Laminate

decision which was then sent back to the simulated autoclave for the next time step. The cycle is repeated for 400 steps (400 simulated minutes). The resulting temperatures were displayed graphically on the computer screen and stored in a file.

Note that the controller was unable to reproduce the temperature hold in the training cycle as the network could not produce a true zero output. However, the final cure cycle obtained was satisfactory and did not lead to any unacceptable condition.

A thicker prepreg with 128-plies was tested (Test 1b) with the same NNC. The results are shown in Figure 4.9. The autoclave temperature increased without slowing down to 450°F, remained constant (as it was the maximum permitted autoclave temperature) for about 30 minutes and then cooled down to 80°F. The cure cycle obtained was not satisfactory for it caused the laminate temperature to exceed 500°F (Tmid>500°F) which was an unacceptable condition.

The controller was then tested (Test 1c) with a 256-ply laminate. The resulting autoclave and laminate temperatures are shown in Figure 4.10. The performance of the Case 1 NNC was even worse than it was in Test 1b. In this test, the laminate temperature exceeded 600°F because of the greater thickness.

Case 2 Training

As the network trained by cure cycle for a 32-ply laminate did not lead to good generalization when tested with laminates of different thicknesses, another training set was created using a thicker panel of 256 plies (Case 2). The training set is shown in Figure 4.11. During the initial heating the laminate top temperature was higher than that

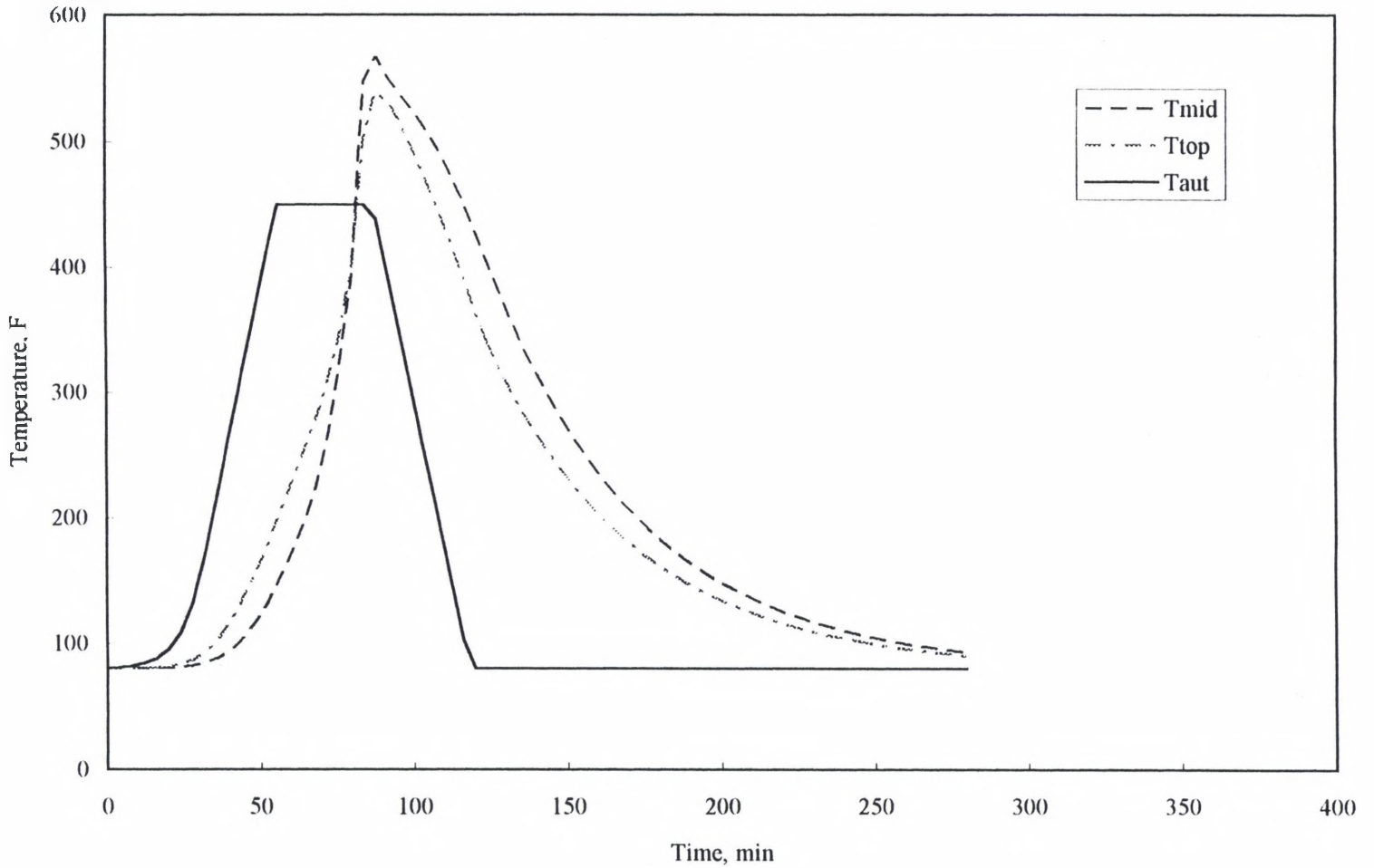


Figure 4.9. Test 1b, Case 1 Trained Network Controlling 128-Ply Laminate

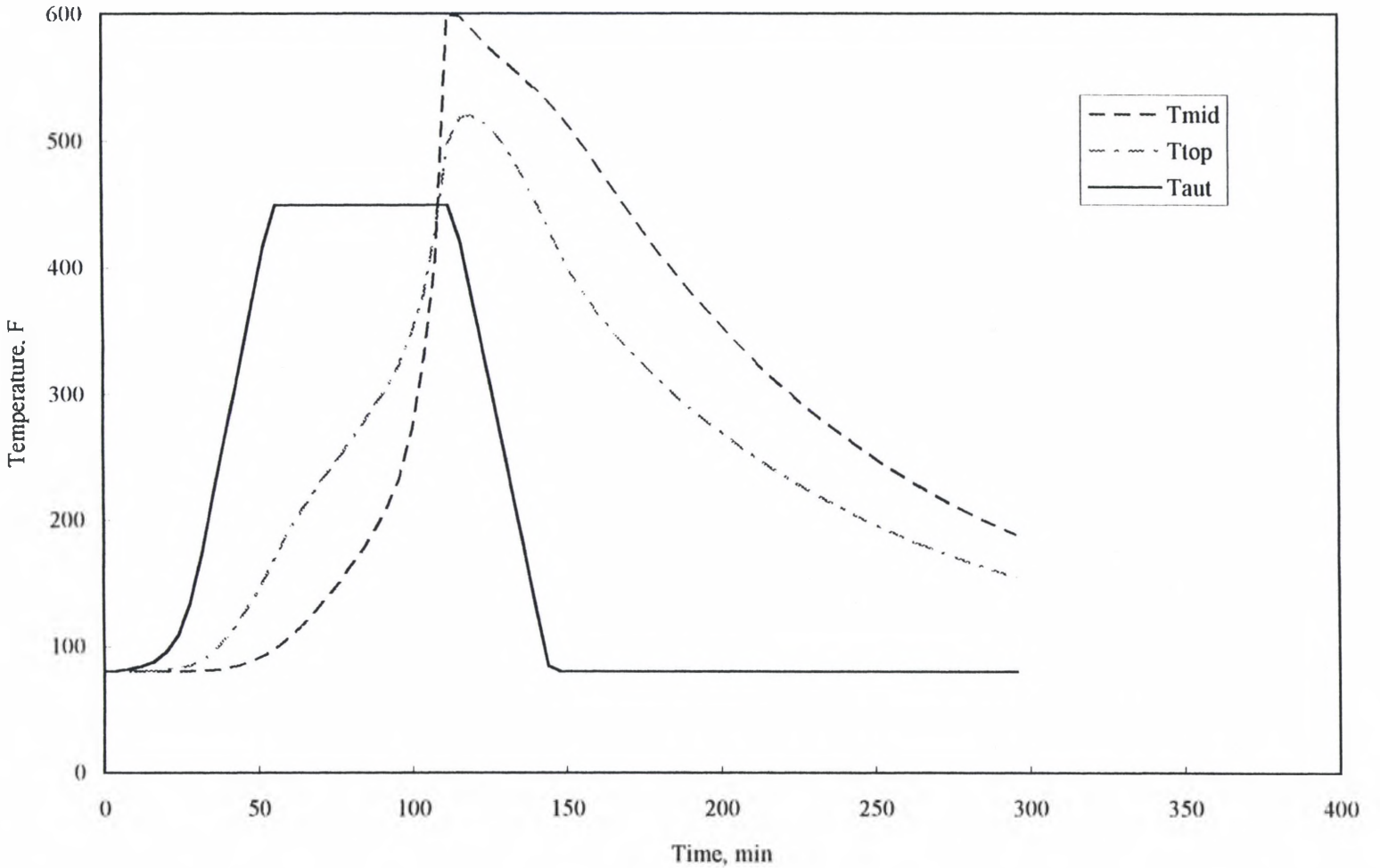


Figure 4.10. Test 1c, Case 1 Trained Network Controlling 256-Ply Laminate

at the midpoint. Cooling started when the midpoint temperature became higher than that of the top (Tmid-Ttop). The RMS errors resulting from various number of hidden neurons are shown in Figure 4.12. They are similar to those of Case 1. A BPNN with one hidden layer containing six neurons was selected for the NNC testing.

Case 2 Testing

Same procedure as in Case 1 was followed for Case 2 testing. Tests 2a, 2b, and 2c were made to control curing of 256-, 32-, and 128-ply laminates, respectively. Test 2a in Figure 4.13, shows that the NNC was able to reproduce the training and control the curing of a 256-ply laminate. The laminate temperature increased slowly but did not exceed 500°F and an acceptable cure cycle was obtained. However, the network was not able to control curing of the 32-ply nor the 128-ply laminate as shown in Figures 4.14 and 4.15. Figure 4.14 shows that, when controlling the curing of a 32-ply laminate, the temperature increased so slowly that the temperature reached only about 120°F after 280 minutes. When controlling the curing of a 128-ply laminate the NNC started cooling before the temperature reached 300°F as can be seen in Figure 4.15.

Case 3 Training

The network was then trained by combining the training sets used in Cases 1 and 2. The RMS errors are shown in Figure 4.16. One hidden layer with six neurons was chosen, as were in Cases 1 and 2, to demonstrate the NNC performance in this case even though the network with seven hidden neurons had the lowest error.

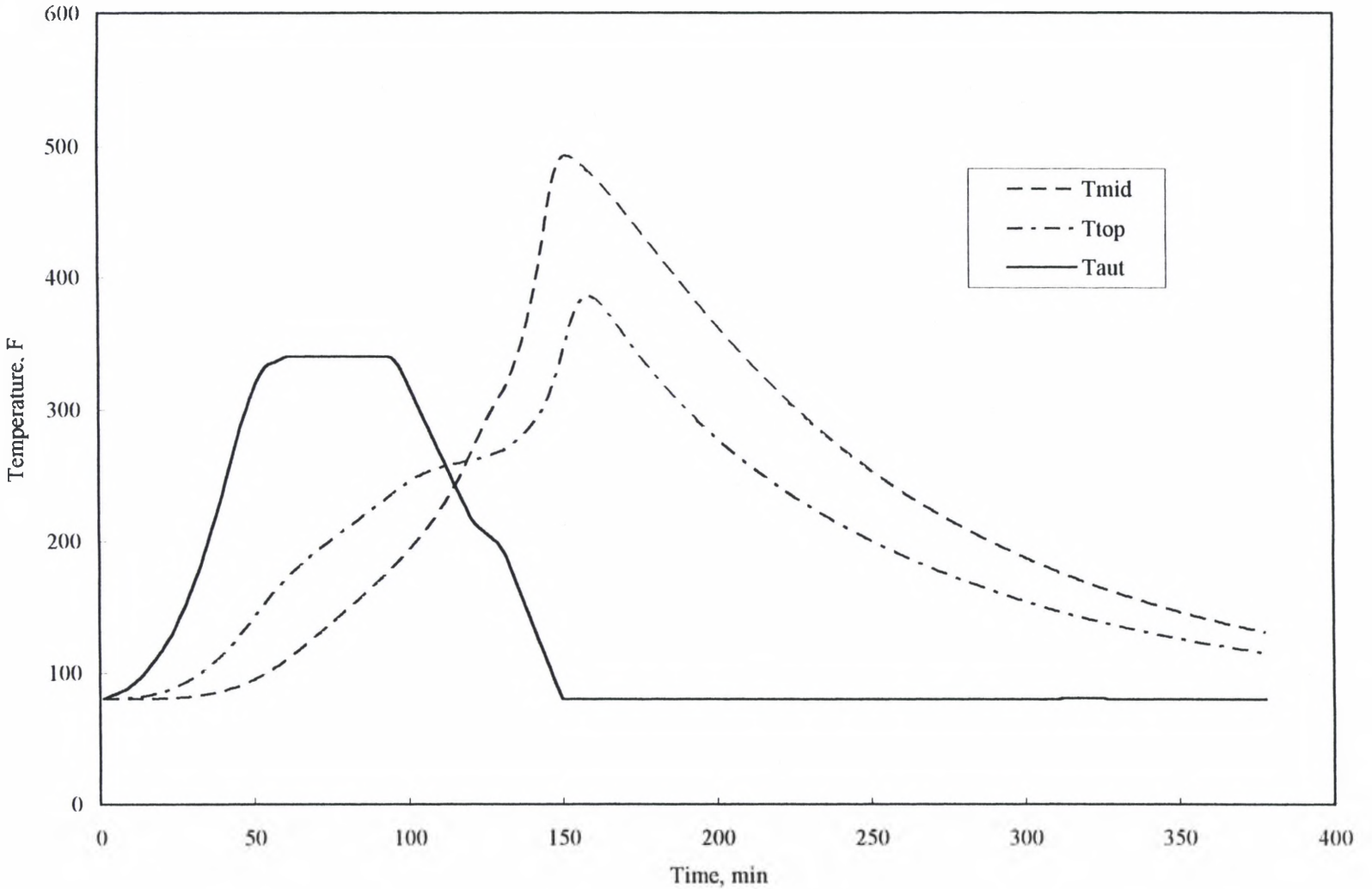


Figure 4.11. Case 2 Training Using Cure Cycle for 256-Ply Laminate

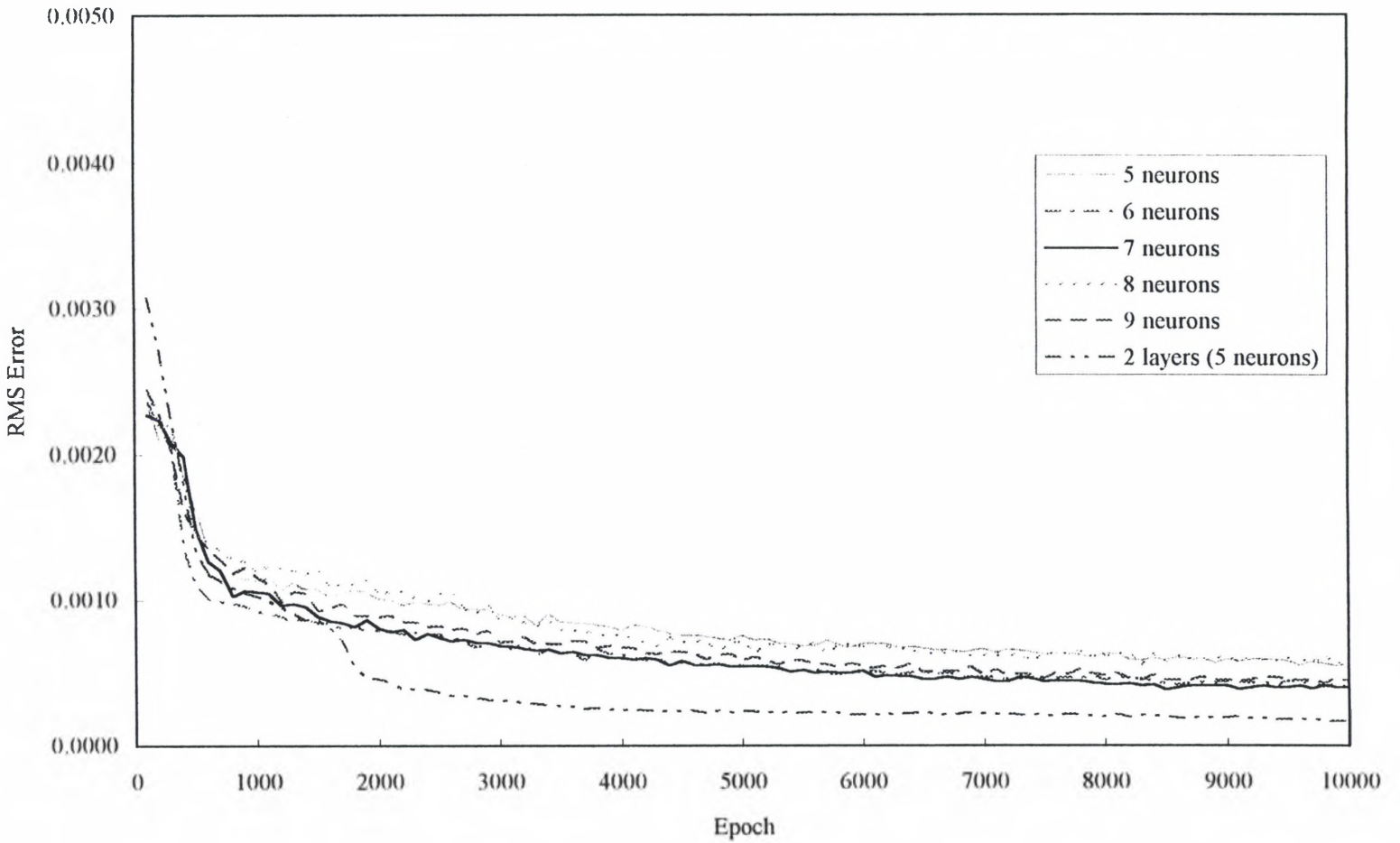


Figure 4.12. Case 2 RMS Error Variation During Training

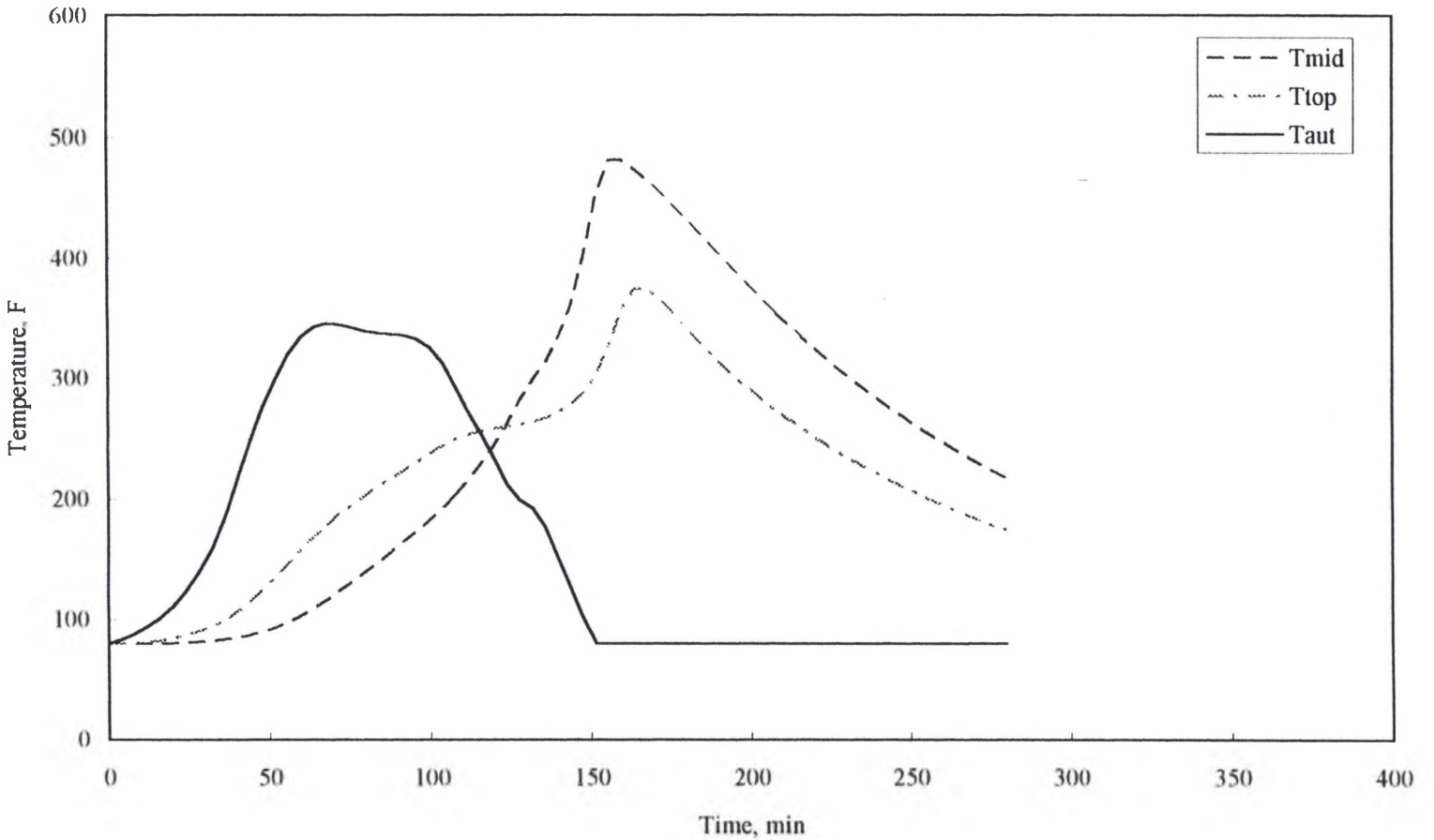


Figure 4.13. Test 2a, Case 2 Trained Network Controlling 256-Ply Laminate

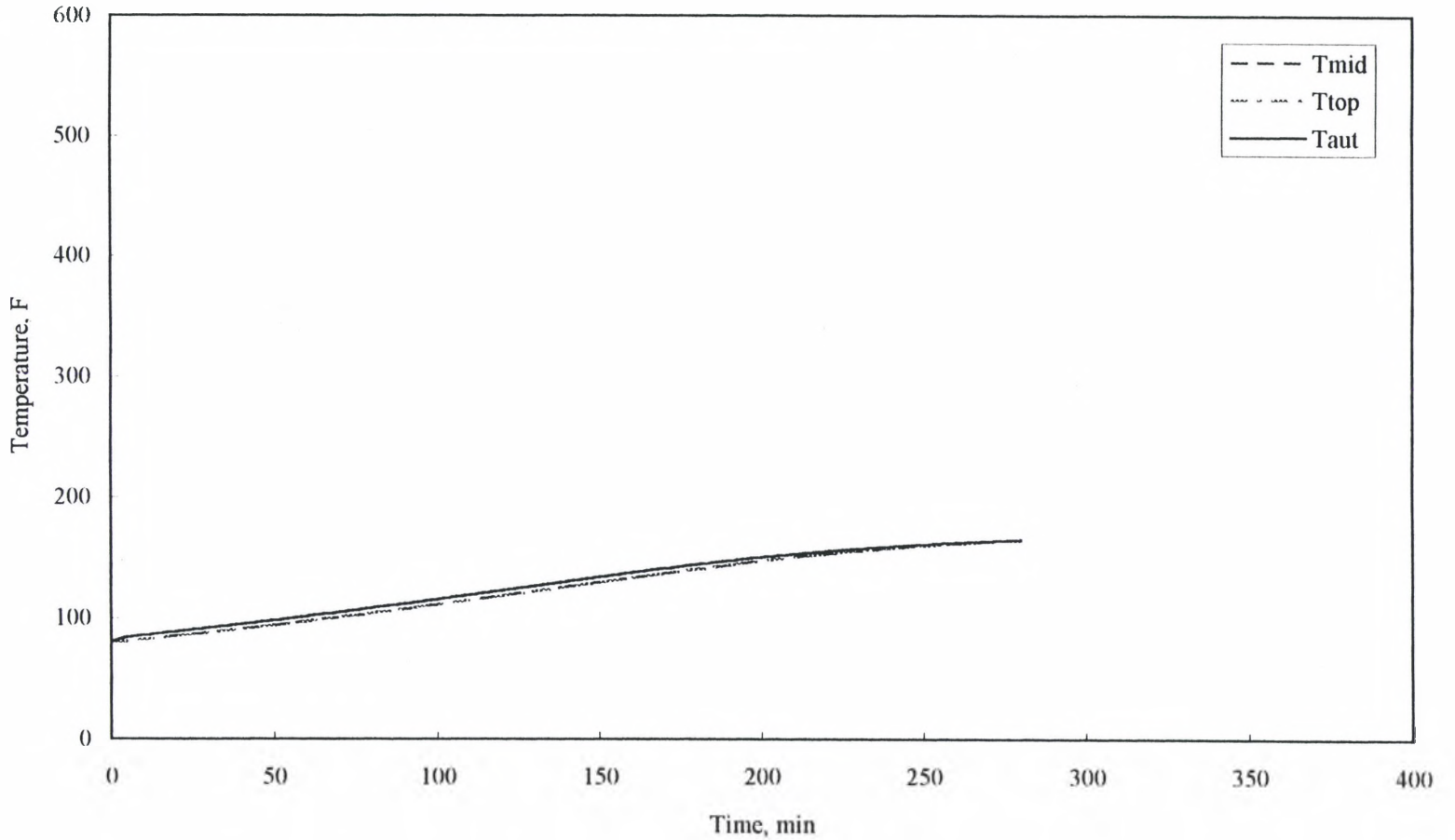


Figure 4.14. Test 2b, Case2 Trained Network Controlling 32-ply Laminate

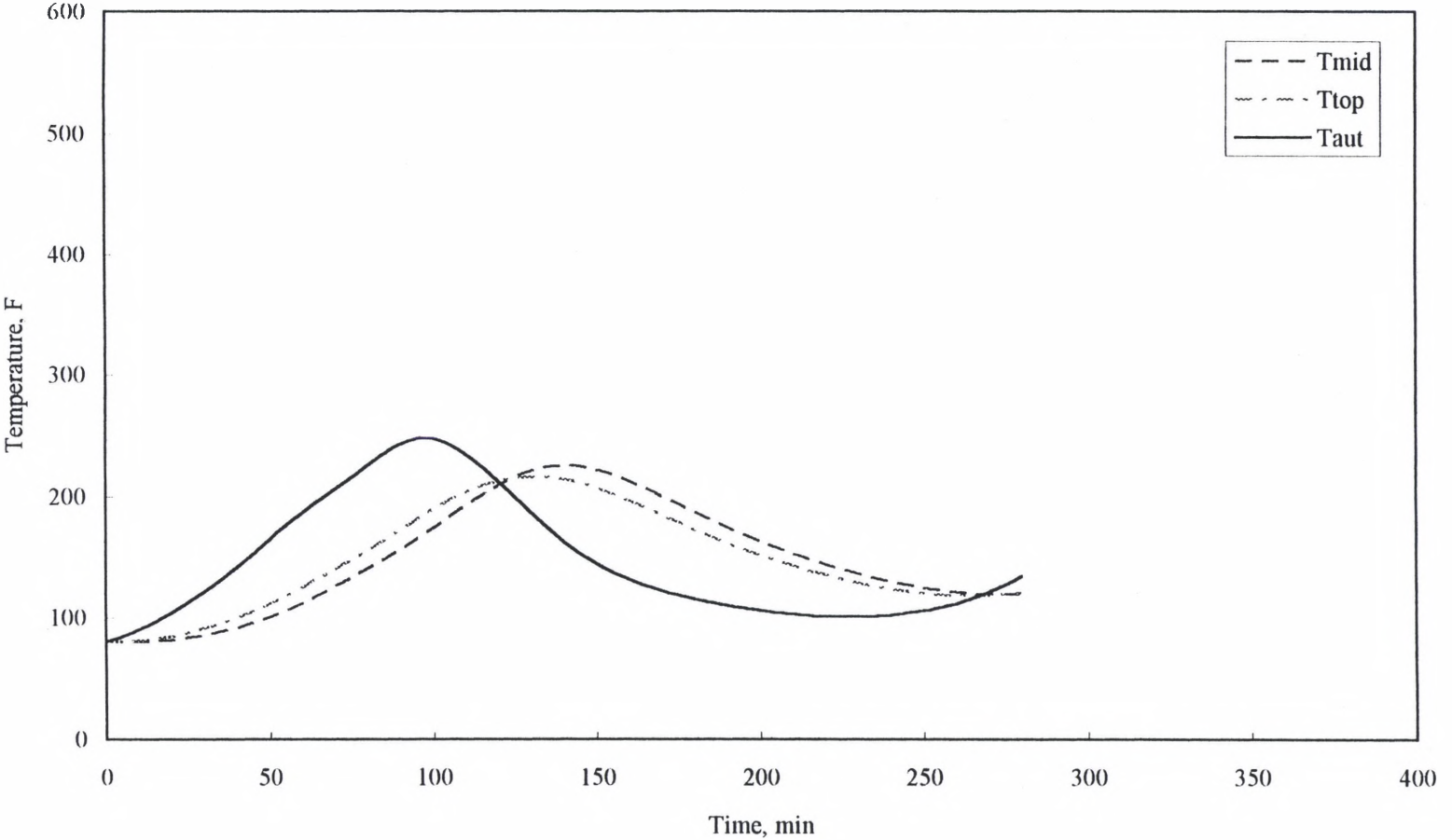


Figure 4.15. Test 2c, Case 2 Trained Network Controlling 128-Ply Laminate

Case 3 Testing

Again, the same test procedure was followed. Tests 3a, 3b, and 3c were made to control the curing of 32-, 128-, and 256-ply laminates, respectively. The NNC successfully controlled the curing of a 32-ply panel in Test 3a, shown in Figure 4.17. The laminate temperature never exceeded 500°F.

Then, an intermediate thickness of 128 plies was used to test the network performance in Test 3b. As can be seen in Figure 4.18, the temperature increased slowly, taking a long time to cure the panel. Although the laminate temperature never exceeded 500°F and the resulting cure cycle would cure the part, the controller behavior was not acceptable as it raised the autoclave temperature while the reaction was rapidly accelerating

Finally, the same network was used to control the curing of a 256-ply panel in Test 3c. It can be seen Figure 4.19, that the laminate temperature went slightly over 500°F. However, the controller performance was considered acceptable. This cure cycle is similar to that in Test 2a, but with generally higher temperatures. The inclusion of the 32-ply training set appeared to have degraded the performance of the NNC.

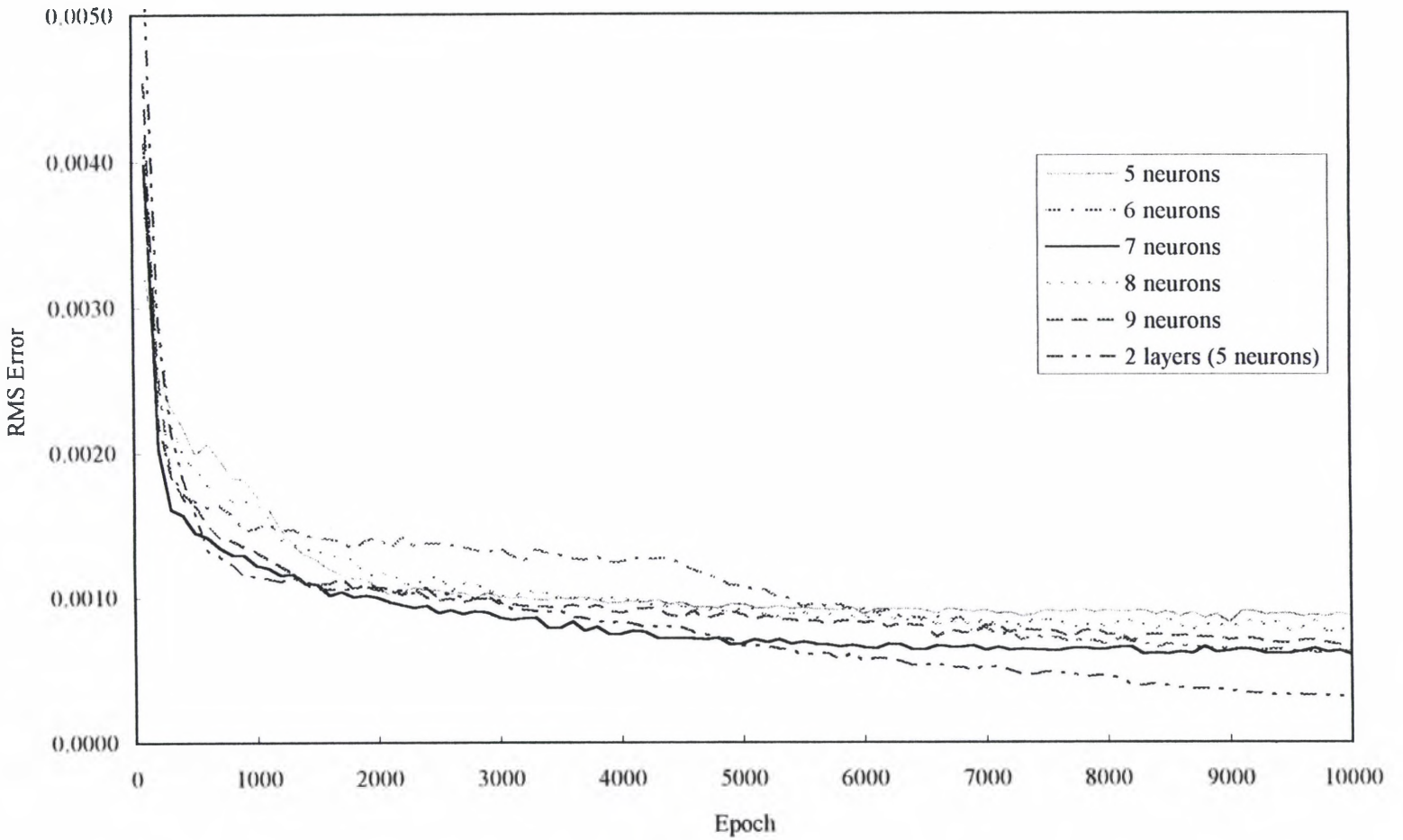


Figure 4.16. Case 3 RMS Error Variation During Training

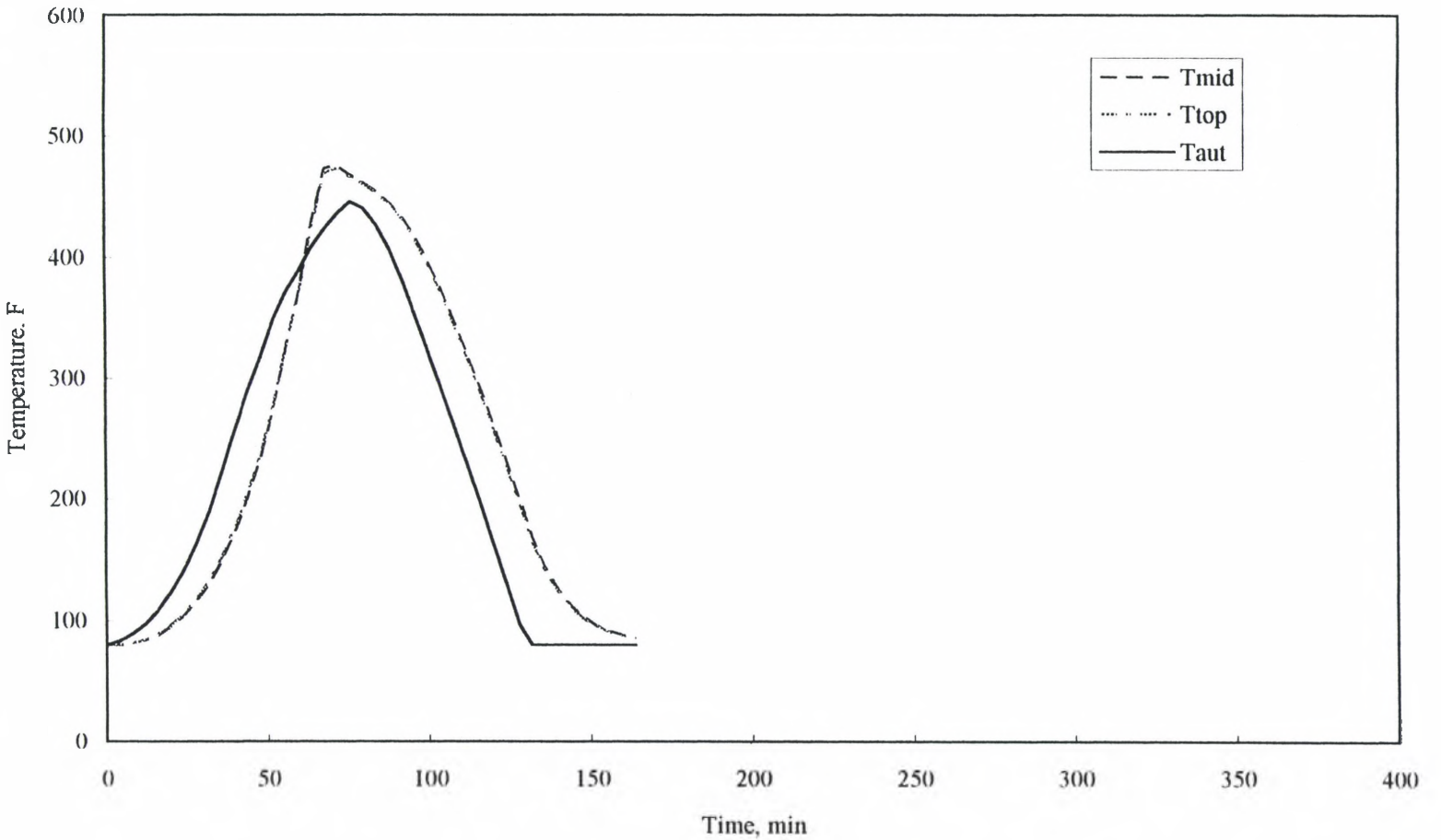


Figure 4.17. Test 3a, Case 3 Trained Network Controlling 32-Ply Laminate

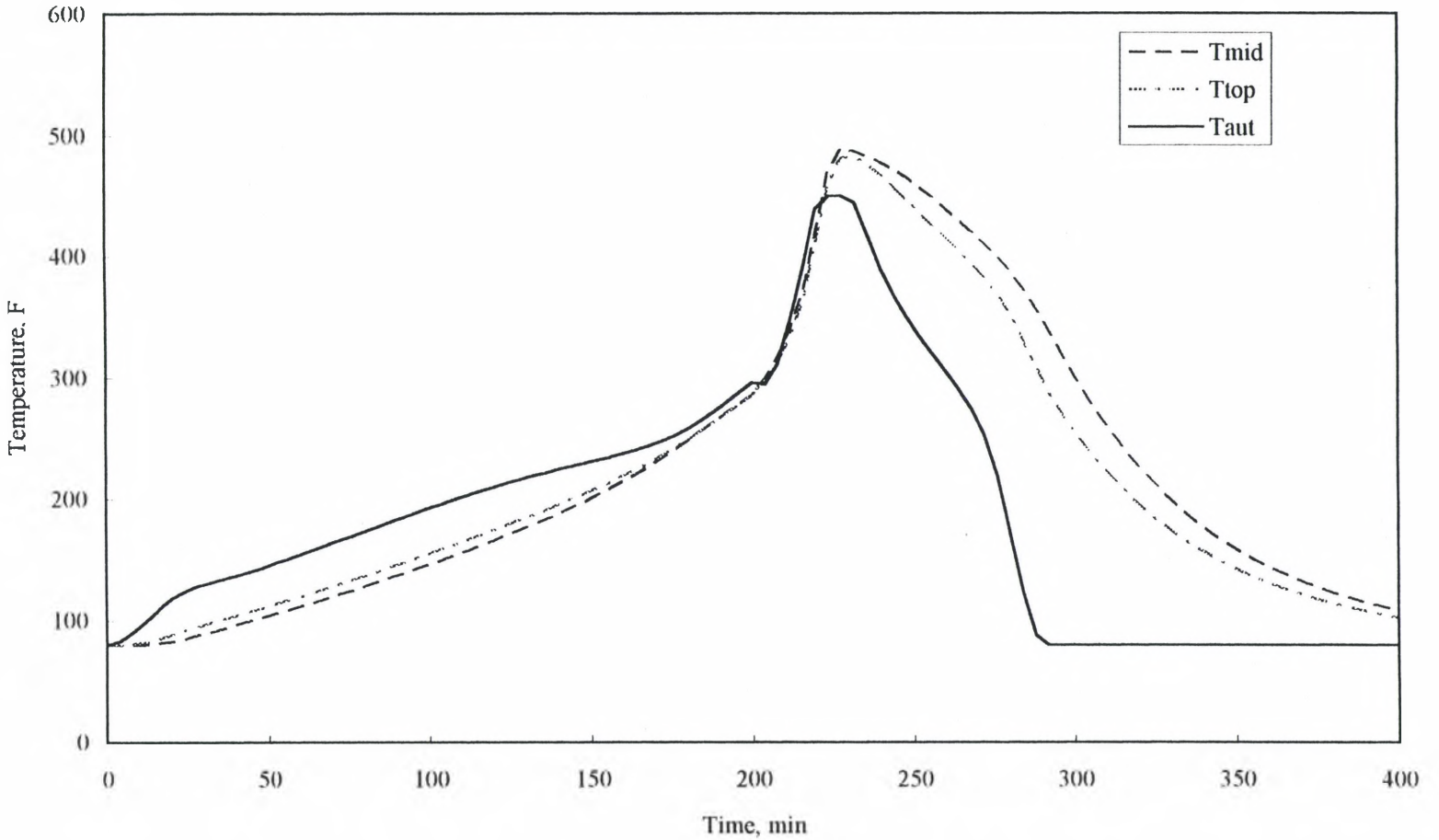


Figure 4.18. Test 3b, Case 3 Trained Network Controlling 128-Ply Laminate

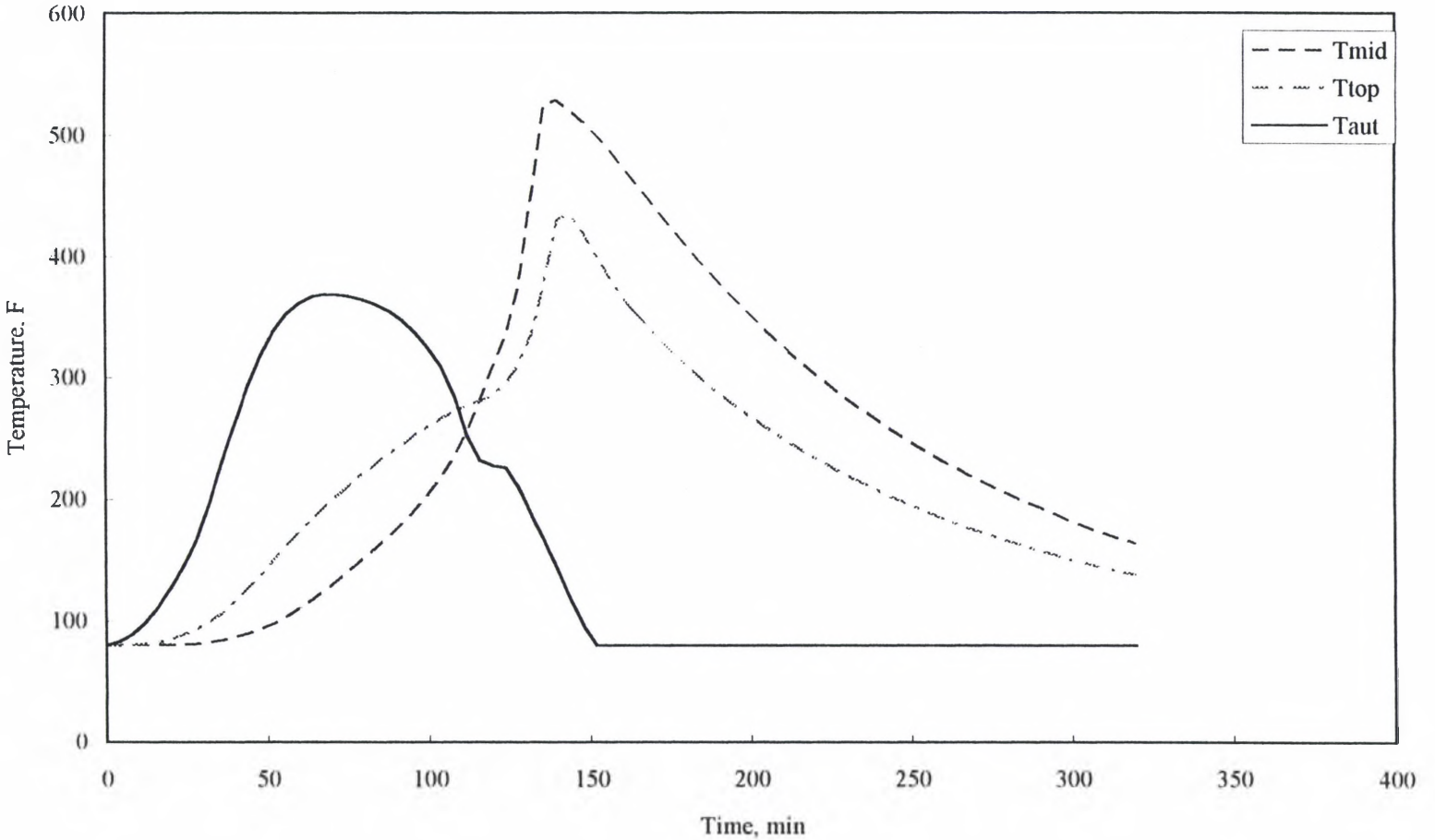


Figure 4.19. Test 3c, Case 3 Trained Network Controlling 256-Ply Laminate

CHAPTER V

SUMMARY AND CONCLUSIONS

A general purpose back propagation neural network (BPNN) has been developed. The network has been trained by operator generated cure cycles to function as a neural network controller (NNC) for self-directed control of a simulated autoclave process of curing of composite material. The NNC obtains temperature data from the simulator, performs computation, and sends a set point adjustment to the simulator for the next execution step. NeuroWindows™, a Windows™ Dynamic Link Library, was used as a tool for building the network under the Visual Basic Development Environment.

NNCs with various number of hidden neurons were studied. Three NNCs with one hidden layer containing six neurons were tested by controlling a simulated autoclave. They were respectively trained by using operator generated cure cycle for a thin 32-ply panel, a thick 256-ply panel, and the combination of those two cure cycles.

The effect of the number of neurons in the hidden layer was interpreted by examining the RMS errors which were calculated based on the desired output values given in the training set and the network predicted output. The comparisons were based on the RMS errors after 10,000 epochs of training. In all cases, the network structure with two hidden layers containg five neurons in each layer resulted in the smallest RMS error. A

summary of the RMS errors is shown in Table 5.1. Although the error comparisons were made at 10,000 training epochs, the NNCs used for testing used link weights after 2,000 training epochs as initial tests indicated overtraining occurred with 10,000 epochs.

Table 5.1

Comparison of RMS Error Obtained After 10,000 Epochs

Number of Neurons in Hidden Layer(s)	Case 1 Trained by 32-Ply Cycle	Case 2 Trained by 256-Ply Cycle	Case 3 Trained by 32- & 256-Ply Cycles
5	0.000661	0.000552	0.000864
6	0.000651	0.000388	0.000616
7	0.000718	0.000392	0.000602
8	0.000864	0.000571	0.000796
9	0.001160	0.000440	0.000660
5, 5 (2 layers)	0.000509	0.000167	0.000310

All test cases are summarized in Table 5.2. From the cases studied it can be seen that the NNC performance was good when the same thickness panel used for the training process was used in the testing. When using the neural network for extrapolation, as in Tests 1b and 1c using Case 1 NNC and in Tests 2b and 2c using Case 2 NNC, the results were not promising. When two training sets were combined to train the Case 3 NNC, it reproduced the training in Tests 3a and 3c but failed to interpolate in Test 3b. Failing to

extrapolate was expected. Failing to interpolate indicated that the network was unable to generalize the control strategy. Although it was surmised that the input vector to the NNC contained all necessary information concerning energy balance of the autoclave curing process, and that the desired output of set point adjustments was based on the operator's understanding of energy balance, there was obviously something missing. One possible cause is the lack of dynamic information to the NNC which was crucial to the operator's decision making. Although one of the inputs, the heat release rate, provides rate information, it carries no history information. Incorporation of temperature rate(s) to the input vector is recommended for future study. One other possible cause is the consistency of the operator's decisions. As there are many paths to achieve the desired outcome, different and conflicting input vectors may be associated with the same set point adjustment decision. This is exemplified by the results of Test 3b. Further examination of the training sets is warranted.

In conclusion, it is believed that a back propagation neural network is capable of capturing an operator's decision mechanism on temperature control of a heat transfer process as it was able to generate a cure cycle having been given only randomized instances of the process state. The five process variables chosen as the input vector can only represent a particular case of heat transfer process and are not enough for the network to generalize. The BPNN tool is flexible enough to accommodate future trials with changes in input vector and hidden layers.

Table 5.2

Summary of the NNC Test Results

NNC Case	Number of Plies for Training	Test	Number of Plies for Testing	Performance Description
1	32	1a	32	Over training observed Good results with 6 neurons in one hidden layer and 2,000 epochs training (Figure 4.8)
1	32	1b	128	Controlled the process but laminate temperature reached 580°F (Figure 4.9)
1	32	1c	256	Controlled the process but laminate temperature reached 600°F (Figure 4.10)
2	256	2a	256	Reproduced temperature cure cycle with good accuracy (Figure 4.13)
2	256	2b	32	Heating was too slow, NNC did not control the process (Figure 4.14)
2	256	2c	128	Heating was too slow, NNC did not control the process (Figure 4.15)
3	32 and 256	3a	32	Over training observed Good results obtained after 8,000 epochs Temperature increased smoothly and control was obtained (Figure 4.17)
3	32 and 256	3b	128	Temperature changed slowly, taking very long time to cure Unacceptable controller behavior, called for heating while reaction accelerating (Figure 4.18)
3	32 and 256	3c	256	Part temperature slightly higher than 500°F but good control obtained (Figure 4.19)

REFERENCES

1. P. R. Ciriscioli and G. S. Springer, "Smart Autoclave Cure of Composites", Technomic Publishing Company, 1990.

2. B. Joseph and F. W. Hanratty, "Predictive Control of Quality in a Batch Manufacturing Process Using Artificial Neural Network Models", *Ind. Eng. Chem. Res.*, 32, 1951-1961, 1993.

3. D. E. Rumelhart and J. L. McClelland (Eds.), "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", MIT Press, Cambridge, Mass, 1986.

4. R. P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-22, April 1987.

5. R. A. Servais, "Process Modeling", University of Dayton, Dayton, OH 1989.

6. R. A. Servais, "Process Modeling and Optimization", Engineered Materials Handbook, Composites, Volume 1, ASM International, Metals Park, Ohio, 1987.

7. G. M. Sheperd, "The Significance of Real Neuron Architectures for Neural Network Simulations", Computational Neuroscience, MIT Press, Cambridge, MA, 1990.

8. H. T Wu and B. Joseph, "Knowledge Based Control of Autoclave Curing of Composites", *SAMPE Journal*, Vol. 26, No. 6, 1990.

9. T. E. Saliba, "Modeling of Heat Transfer in Advanced Composite Materials During the Cure Process", Ph.D. Dissertation, University of Dayton, 1986.

10. J. C. Hoskins and D. M Himmelblau, "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering", *Computers and Chemical Engineering*, Vol. 12, No. 9/10, pp. 881-890, 1988.

11. C. W. Lee and B. P. Rice, "On-Line Determination of Composite Laminate Thermal Diffusivity and Heat Release Rate", 36th International SAMPE Symposium, April 1991.

12. B. P. Rice, "Composite Cure Monitoring with a Tool-Mounted UV-VIS-NIR Fiber Optic Sensor", 39th International SAMPE Symposium, April 1994.

13. G. Ganesh, J. P. Steele, H. Zhang, D. Mishra, and J. Jones, "Predicting Degree-of-Cure of Epoxy Resins with Fiber Optic Sensors and Artificial Neural Networks", 39th International SAMPE Symposium, April 11-14, 1994

14. R. B. Heslehurst, "The Cure Process and Temperature Profiles", International Conference on Advanced Composite Materials , The Minerals, Metals & Materials Society, 1993.

15. J. P. H. Steele, C. Ganesh, K. Liu, H. Zhang, D. Mishra, and J. Jones, "Predicting Degree of Cure of Epoxy Resins Using Dielectric Sensor Data and Artificial Neural Networks", 38th International SAMPE Symposium, May 1993.

16. T. E. Quantrille and Y. A. Liu, "Artificial Intelligence in Chemical Engineering", Academic Press, 1991.

17. M. L. Padgett and T. A. Roppel, "Neural Networks and Knowledge Capture", NASA/CCDS AV SPI, WNN-AIND, 1991.

18. C. W. Lee, "Composite Cure Process Control by Expert System", American Society for Composites, First Technical Conference, Dayton, October 1986.

19. D. A. White and D. A. Sofge, "Handbook of Intelligent Control, Neural, Fuzzy, and Adaptive Approaches", New York, Van Nostrand Reinhold, 1992.

20. P. D. Wasserman, "Advanced Methods in Neural Computing", New York, Van Nostrand Reinhold, 1993.

21. P. D. Wasserman, "Neural Computing", New York, Van Nostrand Reinhold, 1989.

22. G. F. Page, J. B. Gomm, and D. Williams, "Application of Neural Networks to Modelling and Control", London, Chapman & Hall, 1993.

23. P. G. J. Lisboa, "Neural Networks, Current Applications", London, Chapman & Hall, 1992.

24. Engineered Materials Handbook, Composites, Volume 1, ASM International, Metals Park, Ohio, 1987.

25. D. E. Rumelhart, B.Widrow, and M. A. Lehr, "The Basic Ideas in Neural Networks", *Communications of the ACM*, Vol. 37, No. 3, March 1994.

26. N. V. Bhat, P. A. Minderman Jr., T. McAvoy, and N. S. Wang, "Modeling Chemical Process Systems via Neural Computation", *IEEE Control Systems Magazine*, April 1990.

27. H. T. Wu and B. Joseph, "Model Based and Knowledge Based Control of Pultrusion Processes", SAMPE Journal, Vol. 26, No. 6, November/December 1990.

28. V. Venkatasubramanian and K. Chan, "A Neural Network Methodology for Process Fault Diagnosis", AIChE Journal, Vol. 35, No. 12, December 1989.

29. L. Ungar, "A Bioreactor Benchmark for Adaptive Network-based Process Control", Neural Networks for Control, MIT Press, 1990.

30. B. Joseph, F. H. Wang, and D. S.-S. Shieh, "Exploratory Data Analysis: A Comparison of Statistical Methods with Artificial Neural Networks", *Computers and Chemical Engineering*, Vol. 16, No. 4, 1992.

31. A. Kempka, "Activating Neural Networks: Part I", AI Expert, June 1994.

32. M. L. Mavrovouniotis, "Artificial Intelligence in Process Engineering", Academic Press, Inc., San Diego, CA, 1990.

33. M. Guillot , R. Azouzi, and M. C. Cote, "Process Monitoring and Control", Artificial Neural Networks for Intelligent Manufacturing, Chapman & Hall, London, 1994.

34. B. Fernandez, "Performance Analysis of Artificial Neural Networks Methods", Artificial Neural Networks for Intelligent Manufacturing, Chapman & Hall, London, 1994.

35. M. B. Buczek, "Process Control of Advanced Polymeric Composites", Ph.D. Dissertation, University of Dayton, Dayton, OH, July 1994.

36. L. Kalra, M. J. Perry, and L. J. Lee, "Automation of Autoclave Curing of Graphite-Epoxy Composites", *Journal of Composite Materials*, Vol. 26, No. 17, 1992.

37. S. Gustafson and G. Little, Neural Networks Lecture Notes, University of Dayton, 1994.

38. A. C. Loos and G. S. Springer, "Curing of Epoxy Matrix Composites", Journal of Composite Materials, 17, 135-169, 1983.

39. R. J. Henrichs, "Closed-Loop Cure", Engineered Materials Handbook, Composites, Volume 1, ASM International, Metals Park, Ohio, 1987.

40. Microsoft Visual Basic Version 3.0 Manuals, Professional Edition, 1993.

41. R. B. Bird, W. E. Stewart, and E. N. Lightfoot, "Transport Phenomena", John Wiley & Sons, 1960.

42. T. E. Saliba, S. R. Quinter, and F. L. Abrams, "Expert Model for Intelligent Control of Composite Materials processing in a Press", *Composites Engineering*, Vol. 2, No. 2, pp. 105-115, 1992.

43. C. W. Lee et al., "Qualitative Process Automation for Autoclave Cure of Composite Parts", United States Patent, Patent Number: 5,032,525, July 16, 1991.

44. E. F. Phelps, "Challenges for the Composites Industry in the 90's a Global Concern", 39th International SAMPE Symposium, April 11-14, 1994.

APPENDICES

APPENDIX A

Source Code Listing of Neural Network Controller (NNC)

```
' This version supports a single net, with any number of layers

' Declarations

Option Explicit

DefInt I-N
DefSng A-H, O-Z

' General def
Const BLACK = &H0&
Const RED = &HFF&
Const GREEN = &HFF00&
Const YELLOW = &HFFFF&
Const BLUE = &HFF0000
Const MAGENTA = &HFF00FF
Const CYAN = &HFFFF00
Const WHITE = &HFFFFFF
Const GRAY = &H80000008

Dim t$
Dim CrLf$
Dim dummy

' Array Sizes
Const MaxNeuronsPerLayer = 10
Const MaxPatterns = 1000
Const MaxLayers = 10
Const MaxWeights = 200                  ' total =  SumOf [ nNeurons(i)+1) * nNeurons(i+1) ]
                                        '          for i = 0 to nLayers-2


' NN def
Const MomentumBP = 2                    ' BackPropagation with momentum
Const rLearningRate = .7
Const rMomentum = .3                    ' Momentum determines the proportion of the last change
                                        '    added into the new weight change to avoid oscillating
Const WtInitial = .3                    ' Initial weights
Const ScaleInLo = 0!                    ' Scaling ranges for net input and output
Const ScaleInHi = 1!
Const ScaleOutLo = .1
Const ScaleOutHi = .9

Dim NetNumber                           ' Having this shared forces single net for any NeuroWin call
Dim iCode

Dim nLayers, nHiddenLayers
Dim LastLayer, LastHiddenLayer
Dim nNeurons(0 To MaxLayers)
Dim Weights(MaxWeights)

Dim nInputs, nOutputs
```

```
Dim rMinInput(MaxNeuronsPerLayer), rMaxInput(MaxNeuronsPerLayer)
Dim rMinOutput(MaxNeuronsPerLayer), rMaxOutput(MaxNeuronsPerLayer)
Dim InputName$(MaxNeuronsPerLayer), OutputName$(MaxNeuronsPerLayer)

Dim nEpochCount As Long, nEpochCountDefault As Long, LastEpochCount As Long
Dim nSkip, nSkipDefault
Dim ErrorBuffer(MaxPatterns)
Dim LinksFileName$

' flags
Dim HaveCreatedNet As Integer          ' Flag used to limit to a single net at any time
Dim GotLinksFile As Integer
Dim HaveTrainedNet As Integer
Dim TrainMore As Integer
```

```
Function Between (x, xlo, xhi)
   If x < xlo Then
      Between = xlo
   ElseIf x > xhi Then
      Between = xhi
   Else
      Between = x
   End If
End Function
```

```
Static Sub btnLNK_Click (Value As Integer)
Dim LinksCount, b$

      chkRandomIndex.Visible = False
      txtLNK.ForeColor = MAGENTA
      DisableButtons

      ReadLinksFile

      If GotLinksFile Then
         txtLNK.ForeColor = RED
         b$ = txtLNK.Text
         LinksCount = LinksCount + 1
         txtLNK.Text = Parm$(b$, "s") + "s" + Str$(LinksCount)
      Else
         txtLNK.ForeColor = BLACK
      End If


      EnableButtons
```

```
        chkRandomIndex.Visible = True

End Sub

-------------------------------------------------------------------------------------------------------------------------------

Sub btnRunTestFile_Click (Value As Integer)

    If (Not GotLinksFile) And (Not HaveCreatedNet) Then
        btnLNK.Value = True         ' ReadLinksFile
        If Not GotLinksFile Then Exit Sub
    End If

    If Not HaveCreatedNet Then
        CreateNet
        lblNetNumber.Caption = "Net" + Str$(NetNumber)
        PutLinkWeights
    End If

    DisableButtons

    RunTestFile

    EnableButtons

End Sub

-------------------------------------------------------------------------------------------------------------------------------

Sub btnSIM_Click (Value As Integer)

    If Not GotLinksFile Then btnLNK.Value = True

    If Not HaveCreatedNet Then
        CreateNet
        lblNetNumber.Caption = "Net" + Str$(NetNumber)
        PutLinkWeights
    End If

    COMM.PortOpen = True

    DisableButtons
    RunSimulator
    EnableButtons

End Sub

-------------------------------------------------------------------------------------------------------------------------------
```

```
Sub btnTRN_Click (Value As Integer)
Dim b$

   If (Not HaveTrainedNet) Then
      If GotLinksFile Then
         b$ = "You have loaded links from " + LinksFileName$ + "." + CrLf$
         b$ = b$ + "Do you want to start with this LNK and train more?"
         If MsgBox(b$, 4, "TrainNet") = 6 Then
            TrainMore = 1                    ' need to get new TRN and new LNK
         Else
            btnHidden.Value = True
            Exit Sub
         End If
      Else                                   ' no links available either
         TrainMore = 0
      End If
   Else                                      '   HaveTrainedNet
      b$ = "You have just trained a net." + CrLf$
      b$ = b$ + "Do you want to train more?"
      If MsgBox(b$, 4, "TrainNet") = 6 Then
         TrainMore = 2                       ' use the same TRN and LNK
      Else
         btnHidden.Value = True
         Exit Sub
      End If
   End If


   txtNhidden.ForeColor = RED
   txtNrecords.Enabled = False

   DisableButtons

   txtTRN.ForeColor = MAGENTA

   TrainNet

   If HaveTrainedNet Then
      txtTRN.ForeColor = RED
   End If

   EnableButtons
   txtNrecords.Enabled = True

End Sub
```

```
Sub btnViewTRN_Click (Value As Integer)
Dim i, J, nTRN, nPatterns, a$, b$, KeyWord$
Dim Ttop, Tmid, Taut, dTmid, Hr, SPadj
Dim TrainingSetFileName$
ReDim x(MaxPatterns), Y(MaxPatterns)

   OpenFile "Training Set File", "trn", "Input", TrainingSetFileName$, nTRN
   lblTRN.Caption = TrainingSetFileName$
   If TrainingSetFileName$ = "" Then Exit Sub

   frmSIM.Show
   frmSIM.Caption = "Training set data"
   frmSIM.grpTemp.DrawStyle = 1
   frmSIM.grpTemp.GraphStyle = 4
   frmSIM.grpTemp.PatternedLines = 1
   frmSIM.grpTemp.DataReset = 9
   frmSIM.grpTemp.AutoInc = False
   frmSIM.grpTemp.RandomData = False
   frmSIM.grpTemp.NumPoints = 1000
   frmSIM.grpTemp.TickEvery = 200
   frmSIM.grpTemp.LabelEvery = 200
   frmSIM.grpTemp.NumSets = 5

   Do
      dummy = DoEvents()
      If EOF(nTRN) Then Exit Do
      KeyWord$ = InputField$("[]", nTRN)
      Select Case KeyWord$
      Case "INPUT RANGES"                    ' nInputs lines
         i = 0                               '   end this group with a blank line
         Do                                  '   (number of nInputs determined by
            Line Input #nTRN, a$             '     the number of lines that follow)
            If Trim$(a$) = "" Then Exit Do   '   InputNeuronName, lo, hi
            i = i + 1                        '   ...
            InputName$(i) = Trim$(Parm$(a$, t$))
            rMinInput(i) = Val(Parm$(a$, t$))
            rMaxInput(i) = Val(a$)
         Loop
         nInputs = i:          txtNinputs.Text = nInputs
         nNeurons(0) = nInputs

      Case "OUTPUT RANGES"                   ' nOutputs lines
         i = 0                               '   end this group with a blank line
         Do                                  '
            Line Input #nTRN, a$             '   OutputNeuronName, lo, hi
            If Trim$(a$) = "" Then Exit Do   '   ...
            i = i + 1
            OutputName$(i) = Trim$(Parm$(a$, t$))
            rMinOutput(i) = Val(Parm$(a$, t$))
            rMaxOutput(i) = Val(a$)
         Loop
```

```
    nOutputs = i:          txtNoutputs.Text = nOutputs

    ' Hidden Layers from screen input
    a$ = txtNhidden.Text
    b$ = ""
    For i = 1 To MaxLayers - 1
       nNeurons(i) = Val(Parm$(a$, ","))
        b$ = b$ + Str$(nNeurons(i)) + ","
       If a$ = "" Then Exit For
    Next:                  txtNhidden.Text = Left$(Trim$(b$), Len(b$) - 2)
    nHiddenLayers = i:
    nLayers = nHiddenLayers + 2
    LastHiddenLayer = i
    LastLayer = i + 1
    nNeurons(LastLayer) = nOutputs

  Case "TRAINING DATA"                    ' (nInputs + nOutputs) pieces of data
    J = 0                                 '      program reads to the end of file
    Do                                    '      counter gives number of traing patterns
       dummy = DoEvents()

       If EOF(nTRN) Then Exit Do
       Line Input #nTRN, a$
       "Text.Text = Str$(J) + " " + a$
       If Trim$(a$) <> "" Then
          J = J + 1
          For i = 1 To nInputs          ' Mapping input vector to xMapped(i,j) and yMapped(i,j)
             x(i) = Val(Parm$(a$, t$))
          Next
          For i = 1 To nOutputs
             Y(i) = Val(Parm$(a$, t$))
          Next
          Tmid = x(1)
          Ttop = x(1) - x(2)
          Taut = x(1) - x(3)
          "dTmid = X(4)
          Hr = x(5)
          SPadj = Y(1)
          frmSIM.grpTemp.ThisPoint = J
          frmSIM.grpTemp.ThisSet = 1
          frmSIM.grpTemp.GraphData = Tmid
          frmSIM.grpTemp.ThisSet = 2
          frmSIM.grpTemp.GraphData = Ttop
          frmSIM.grpTemp.ThisSet = 3
          frmSIM.grpTemp.GraphData = Taut
          frmSIM.grpTemp.ThisSet = 4
          frmSIM.grpTemp.GraphData = Hr * 10
          frmSIM.grpTemp.ThisSet = 5
          frmSIM.grpTemp.GraphData = SPadj * 10
       End If
    Loop
```

```
            nPatterns = J:      txtNpatterns.Text = Str$(nPatterns)
            Exit Do
        End Select
    Loop
    Close #nTRN

    frmSIM.grpTemp.GraphTitle = TrainingSetFileName$
    frmSIM.grpTemp.NumPoints = (nPatterns \ 50 + 1) * 50
    frmSIM.grpTemp.TickEvery = frmSIM.grpTemp.NumPoints / 5
    frmSIM.grpTemp.LabelEvery = frmSIM.grpTemp.TickEvery
    frmSIM.grpTemp.DrawMode = 2

    btnHidden.Value = True

End Sub

-----------------------------------------------------------------------------------------------------------------------------------


Sub cmdQUIT_Click ()
    iCode = KillNet(NetNumber)
    Close
    End
End Sub

-----------------------------------------------------------------------------------------------------------------------------------


Sub cmdReset_Click ()

    Close
    iCode = KillNet(NetNumber)
    NetNumber = -1
    HaveCreatedNet = False
    GotLinksFile = False

    nEpochCount = nEpochCountDefault
    txtEpochCount = Trim$(Str$(nEpochCount))
    nSkip = nSkipDefault
    txtNrecords = Trim$(Str$(nSkip))

    grpErrors.DataReset = 9
    grpErrors.NumPoints = 2
    grpErrors.GraphData = 0
    grpErrors.BottomTitle = "Epochs/ " + Str$(nSkip)
    grpErrors.Refresh
    grpErrors.DrawMode = 3

    grpLinks.DataReset = 9
    grpLinks.NumPoints = 2
    grpLinks.GraphData = 0
```

```
    grpLinks.Refresh
    grpLinks.DrawMode = 3

End Sub

---

Sub CreateNet ()
' Creates a Momentum BP NN of any number of layers [nLayers]
' with any number of neurons in each single-slab layer [nNeurons()].
' nLayers and nNeurons() must be specified and shared
' NetNumber is established by the Sub

Dim n

    If HaveCreatedNet Then                    ' Current version allows a single net
        Beep
        MsgBox "Net already exist.    A new net is created.", 16, "CREATE NET"
    End If

    ' Define the network number
    iCode = GetNextNet(NetNumber):    If iCode Then GoTo ErrorCreateNet

    ' Make BackPropagation with momentum and enter the serial number
    iCode = MakeNet(NetNumber, MomentumBP, 645213508):    If iCode Then GoTo ErrorCreateNet

    ' Build nLayer slabs in the network, (each layer has a single slab)
    iCode = MakeSlab(NetNumber, 0, nNeurons(0), InputLayer%):    If iCode Then GoTo ErrorCreateNet
    For n = 1 To LastHiddenLayer
        iCode = MakeSlab(NetNumber, n, nNeurons(n), HiddenLayer%):    If iCode Then GoTo
ErrorCreateNet
    Next
    iCode = MakeSlab(NetNumber, LastLayer, nNeurons(LastLayer), OutputLayer%):    If iCode Then
GoTo ErrorCreateNet

    ' Link slabs together (allocates memory to contain the weights)
    ' Creates link n between slabs n and n+1
    ' Initializes the weights to between +/- WtInitial
    For n = 0 To LastHiddenLayer
        iCode = MakeLink(NetNumber, n, WtInitial, n, n + 1):    If iCode Then GoTo ErrorCreateNet
    Next

    HaveCreatedNet = True
    lblNetNumber.Caption = "Net#" + Trim$(Str$(NetNumber))
    Text.Text = Text.Text + "Net#" + Trim$(Str$(NetNumber))
Exit Sub
```

```
ErrorCreateNet:
   MsgBox NetError$(iCode), 16, "Error in CreateNet"

End Sub

---

Sub DisableButtons ()

   btnTRN.Enabled = False
   btnLNK.Enabled = False
   btnRunTestFile.Enabled = False
   btnSIM.Enabled = False

End Sub

---

Sub EnableButtons ()

   btnTRN.Enabled = True
   btnLNK.Enabled = True
   btnRunTestFile.Enabled = True
   btnSIM.Enabled = True

   btnHidden.Value = True

End Sub

---

Sub Form_Load ()
Dim i, a$

   t$ = Chr$(9)
   CrLf$ = Chr$(13) + Chr$(10)

   HaveCreatedNet = False
   HaveTrainedNet = False
   GotLinksFile = False
   NetNumber = -1

   nEpochCount = txtEpochCount.Text
   nEpochCountDefault = txtEpochCount.Text
   nSkip = txtNrecords.Text
   nSkipDefault = txtNrecords.Text
   grpErrors.BottomTitle = "Epochs/ " + Str$(nSkip)
```

```
    ' Communication definitions
    COMM.CommPort = Val(txtCOMMport.Text)
    COMM.Settings = "9600,N,8,1"
    COMM.InputLen = 0
    COMM.OutBufferCount = 0
    COMM.InBufferCount = 0

End Sub
```

```
Sub GetLinkWeights ()
Dim J, k, m, n

    'iCode = Getlink(NetNumber, 0, Weights(0)):    If iCode Then GoTo ErrorGetLinkWeights
    'For n = 1 To LastHiddenLayer
    '    j = (nNeurons(n - 1) + 1) * nNeurons(n)
    '    iCode = Getlink(NetNumber, n, Weights(j)):    If iCode Then GoTo ErrorGetLinkWeights
    'Next

     m = 0
     For n = 0 To LastLayer - 1          ' links 0, 1, ...
        For J = 0 To nNeurons(n)
           For k = 1 To nNeurons(n + 1)
              iCode = GetWeight(NetNumber, n, J, k, Weights(m)):
                 If iCode Then GoTo ErrorGetLinkWeights
              m = m + 1
           Next
        Next
     Next

Exit Sub

ErrorGetLinkWeights:
    MsgBox NetError$(iCode), 16, "Error in GetLinkWeights"

End Sub
```

```
Function GreaterOf (a, b)
    If a > b Then
       GreaterOf = a
    Else
       GreaterOf = b
    End If
End Function
```

```
Sub grpErrors_DblClick ()
   grpErrors.PrintStyle = 1
   grpErrors.DrawMode = 5
End Sub
```

```
Sub grpLinks_DblClick ()
   grpLinks.PrintStyle = 1
   grpLinks.DrawMode = 5
End Sub
```

```
Function InputField$ (Delimiters$, nFileHandle)
' read from nFileHandle
' Skips lines to specified Delimiters$
' Returns the enclosed text in upper case

Dim L$, R$, a$

   L$ = Left$(Delimiters$, 1)
   R$ = Right$(Delimiters$, 1)

   Do
      Line Input #nFileHandle, a$
      a$ = LTrim$(a$)
      If Left$(a$, 1) = L$ Then
         InputField$ = UCase$(Mid$(a$, 2, InStr(a$, R$) - 2))
         Exit Do
      End If
   Loop

End Function
```

```
Sub lblEpochs_Click ()
   txtEpochCount.Text = 10000
   nEpochCount = 10000
End Sub
```

```
Function Map! (ByVal Value!, min!, max!, x1!, x2!)
' Map variables to the range 0 - 1

   If min! = max! Then
      MsgBox "Min and max are equal in Map: " + Str$(min!) + ", " + Str$(max!), 16, ""
      Map! = 0!
      End
      Exit Function
   End If
   If Value! < min! Then Value! = min!
   If Value! > max! Then Value! = max!
   Map! = (x2! - x1!) * (Value! - min!) / (max! - min!) + x1!

End Function

Function NewName$ (GivenName$, NewExt$)
' Change file extension of GivenName$ to NewExten$sion

Dim i

   i = InStr(GivenName$, ".")
   If i <> 0 Then
      NewName$ = Left$(GivenName$, i - 1) + "." + NewExt$
   Else
      NewName$ = GivenName$
   End If

End Function

Function nTotalWeights (nL, nn())
Dim i, n

   n = 0
   For i = 0 To nL - 2                    ' nL is number of layers
      n = n + (nn(i) + 1) * nn(i + 1)     ' nN(i) is the number of neurons in layer i
   Next
   nTotalWeights = n

End Function
```

```
Sub OpenFile (Descript$, ext$, Action$, TheFile$, nFileHandle)
' Get file title from GivenName and construct NewFileName$ with Ext$
' Action is "A" for Append, "O" for Output, else for Input
' Assumes CMDialog object with the name diaFiles

Dim Act$

   On Error GoTo ErrorOpenFile
```

```
    diaFiles.DialogTitle = "Select " + Descript$ + " name for " + Action$
    diaFiles.Filter = "All Files(*.*)|*.*|"
    diaFiles.Filter = diaFiles.Filter + Descript$ + " (*." + ext$ + ")|*." + ext$ + "|"
    diaFiles.FilterIndex = 2
    Act$ = UCase$(Left$(Action$, 1))
    If Act$ = "O" Or Act$ = "A" Then
       diaFiles.Action = 2
    Else
       diaFiles.Action = 1
    End If
    TheFile$ = diaFiles.Filename
    diaFiles.Filename = ""
    nFileHandle = FreeFile
    If Act$ = "O" Then
       Open TheFile$ For Output As #nFileHandle
    ElseIf Act$ = "A" Then
       Open TheFile$ For Append As #nFileHandle
    Else
       Open TheFile$ For Input As #nFileHandle
    End If

    dummy = DoEvents()

Exit Sub

ErrorOpenFile:
    If Not diaFiles.CancelError Then
       MsgBox "Error in OpenFile", 16, ""
    End If
    Resume QuitOpenFile

QuitOpenFile:
    TheFile$ = ""

End Sub
```

```
Function Parm$ (b$, Delimiter$)
' Extract string to the left of the Delimiter$ as Parm$
' Input string b$ is truncated

Dim i

    i = InStr(b$, Delimiter$)
    If i <> 0 Then
       Parm$ = Left$(b$, i - 1)
       b$ = Right$(b$, Len(b$) - i)
       b$ = Trim$(b$)
    Else
```

```
    Parm$ = b$
    b$ = ""
  End If

End Function
```

```
Sub PrintLinksFileHeader (nLNK, nPatterns, LNKdescription$)
ReDim NNname$(MaxLayers, MaxNeuronsPerLayer)
Dim i, J, n

  Print #nLNK, "[This File] ":          Print #nLNK, LinksFileName$ + ", " + LNKdescription$
  Print #nLNK,
  Print #nLNK, "[Training Set] ":       Print #nLNK, lblTRN.Caption
  Print #nLNK,
  Print #nLNK, "[Patterns]":            Print #nLNK, nPatterns
  Print #nLNK,
  Print #nLNK, "[Learning Rate]":       Print #nLNK, rLearningRate
  Print #nLNK,
  Print #nLNK, "[Momentum]":            Print #nLNK, rMomentum
  Print #nLNK,
  Print #nLNK, "[Layers]":              Print #nLNK, nLayers
  Print #nLNK,
  Print #nLNK, "[Neurons]"
        For n = 0 To LastLayer
          Print #nLNK, nNeurons(n); t$;
        Next: Print #nLNK,
  Print #nLNK,
  Print #nLNK, "[Input Ranges]"
        For i = 1 To nInputs
          Print #nLNK, InputName$(i); t$; rMinInput(i); t$; rMaxInput(i)
        Next
  Print #nLNK,
  Print #nLNK, "[Output Ranges]"
        For i = 1 To nOutputs
          Print #nLNK, OutputName$(i); t$; rMinOutput(i); t$; rMaxOutput(i)
        Next
  Print #nLNK,

  Print #nLNK, "[Weights]"
        Print #nLNK, " "; t$;          ' header 1
        For i = 0 To nTotalWeights(nLayers, nNeurons()) - 1
          Print #nLNK, i; t$;
        Next: Print #nLNK,
        Print #nLNK, " "; t$;          ' header 2, weights identifier
        For n = 0 To LastLayer
          NNname$(n, 0) = "Bias" + Trim$(Str$(n))
          For J = 1 To nNeurons(n)
            If n = 0 Then
```

```
            NNname$(n, J) = InputName$(J)
          ElseIf n = LastLayer Then
            NNname$(n, J) = OutputName$(J)
          Else
            NNname$(n, J) = "H" + Trim$(Str$(n)) + "." + Trim$(Str$(J))
          End If
        Next
      Next
      For n = 0 To LastHiddenLayer
        For i = 0 To nNeurons(n)
          For J = 1 To nNeurons(n + 1)
            Print #nLNK, NNname$(n, i) + "_" + NNname$(n + 1, J) + t$;
          Next
        Next
      Next   ' layer
      Print #nLNK,

End Sub

---------------------------------------------------------------------------------------------------------------------------------


Sub PutLinkWeights ()
Dim J, k, m, n

'   iCode = PutLink(NetNumber, 0, Weights(0)):   If iCode Then GoTo ErrorPutLinkWeights
'   For n = 1 To LastHiddenLayer
'      j = (nNeurons(n - 1) + 1) * nNeurons(n)
'      iCode = PutLink(NetNumber, n, Weights(j)):   If iCode Then GoTo ErrorPutLinkWeights
'   Next

    m = 0
    For n = 0 To LastLayer - 1                  ' links 0, 1, ...
      For J = 0 To nNeurons(n)
        For k = 1 To nNeurons(n + 1)
          iCode = PutWeight(NetNumber, n, J, k, Weights(m)):   If iCode Then GoTo
ErrorPutLinkWeights
          m = m + 1
        Next
      Next
    Next

Exit Sub

ErrorPutLinkWeights:
  MsgBox NetError$(iCode), 16, "Error in PutLinkWeights"

End Sub

---------------------------------------------------------------------------------------------------------------------------------
```

```
Sub RandomIndex (Limit, IndexArray())
Dim i, MaxIndex, Index, iHold

   Randomize Timer

   For i = 1 To Limit
      MaxIndex = Limit - i + 1
      Index = Int(Rnd * (MaxIndex)) + 1
                  '''(1 <= rnd num <= MaxIndex)
      iHold = IndexArray(Index)
      IndexArray(Index) = IndexArray(MaxIndex)
      IndexArray(MaxIndex) = iHold
   Next

End Sub

----------------------------------------------------------------------------------------------------------------


Sub ReadLinksFile ()                          ' Read and view links
Dim nEp As Long, nEp1 As Long
Dim i, n, nLNKin, nWts, nCount, AvgErr
Dim a$, b$, KeyWord$, LNKdescription$

   GotLinksFile = False

   '''''''''''''''''''''''''''''''''''''''''''''''''''   Open and Read Links File
   If TrainMore = 2 Then
      nLNKin = FreeFile
      b$ = lblLNK.Caption
      Open Parm$(b$, ",") For Input As #nLNKin
   Else
      OpenFile "Links File", "lnk", "Input", LinksFileName$, nLNKin ' All parameters are tab delimited
      If LinksFileName$ = "" Then Exit Sub
      lblLNK.Caption = LinksFileName$
   End If
   Do
      If EOF(nLNKin) Then Exit Do
      dummy = DoEvents()
      KeyWord$ = InputField$("[]", nLNKin)

      Select Case KeyWord$
      Case "THIS FILE"                        ' 1 line, file name of training set
         Line Input #nLNKin, LNKdescription$
         lblLNK.Caption = LNKdescription$

      Case "TRAINING SET"                     ' 1 line, file name of training set
         Line Input #nLNKin, a$
         Text.Text = "Trained by " + a$
```

```
    Case "PATTERNS"                 ' 1 line, 1 INTEGER
      Line Input #nLNKin, a$
      txtNpatterns.Text = Val(a$)

    Case "LEARNING RATE"            ' 1 line, 1 SINGLE
      Line Input #nLNKin, a$        '   ignored

    Case "MOMENTUM"                 ' 1 line, 1 SINGLE
      Line Input #nLNKin, a$        '   ignored

    Case "LAYERS"                   ' 1 line, 1 INTEGER
      Line Input #nLNKin, a$        '   must precede [NEURONS]
      nLayers = Val(a$)
      nHiddenLayers = nLayers - 2
      LastLayer = nLayers - 1
      LastHiddenLayer = LastLayer - 1

    Case "NEURONS"                  '1 line, nLayers INTEGERs
      Line Input #nLNKin, a$        '   number of neurons for each layer
      For n = 0 To LastLayer        '   must preced [INPUT RANGES] and [OUTPUT RANGES]
        nNeurons(n) = Val(Parm$(a$, t$))
      Next
      nInputs = nNeurons(0)
              txtNinputs.Text = Str$(nInputs)
      b$ = ""
      For i = 1 To nHiddenLayers
        b$ = b$ + Str$(nNeurons(i)) + ","
      Next
              txtNhidden.Text = Left$(Trim$(b$), Len(b$) - 2)
      nOutputs = nNeurons(LastLayer)
              txtNoutputs.Text = Str$(nOutputs)

      nWts = nTotalWeights(nLayers, nNeurons())

    Case "INPUT RANGES"             ' nInputs lines
      For i = 1 To nInputs          '   InputNeuronName, lo, hi
        Line Input #nLNKin, a$      '   ...
        InputName$(i) = Trim$(Parm$(a$, t$))
        rMinInput(i) = Val(Parm$(a$, t$))
        rMaxInput(i) = Val(Parm$(a$, t$))
      Next

    Case "OUTPUT RANGES"            ' nOutputs lines
      For i = 1 To nOutputs         '   OutputNeuronName, lo, hi
        Line Input #nLNKin, a$      '   ...
        OutputName$(i) = Trim$(Parm$(a$, t$))
        rMinOutput(i) = Val(Parm$(a$, t$))
        rMaxOutput(i) = Val(Parm$(a$, t$))
      Next
```

```
    Case "WEIGHTS"                       ' Reads to EOF, ignoring blank lines
      Line Input #nLNKin, a$             '    header 1
      Line Input #nLNKin, a$             '    header 2
                                         '    Input Vector... , Output Vector...
      nCount = 0
      Do
        dummy = DoEvents()
        If EOF(nLNKin) Then Exit Do
        Line Input #nLNKin, a$
        If Trim$(a$) <> "" Then
          nEp = Val(Parm$(a$, t$)):          txtEpochs.Text = nEp
          nCount = nCount + 1
          If nCount = 2 Then
            nEp1 = nEp
          ElseIf nCount = 3 Then
            nSkip = nEp - nEp1:              txtNrecords.Text = Str$(nSkip)
          End If
          b$ = "nEpochs: " + Str$(nEp) + ": "
          For i = 0 To nWts - 1
            Weights(i) = Val(Parm$(a$, t$))
            b$ = b$ + Str$(Weights(i)) + " / "
          Next:
          If chkDisplay.Value Then Text.Text = b$

          AvgErr = Val(Parm$(a$, t$)):       lblGeneral.Caption = "AvgErr=" + Str$(AvgErr)
          ErrorBuffer(nCount) = AvgErr
        End If
      Loop While nEp < nEpochCount
      nEpochCount = nEp:                     txtEpochCount.Text = Str$(nEpochCount)
      LastEpochCount = nEp
      Exit Do
  End Select
Loop

Close #nLNKin

''''''''''''''''''''''''''''''''''''''''''''''''''''''''  View AvgErr Data
grpErrors.DataReset = 9                   '     ...
grpErrors.NumPoints = (nCount \ 25 + 1) * 25
grpErrors.BottomTitle = "Epochs /" + Str$(nSkip)
grpErrors.GraphTitle = "AvgErr (" + LinksFileName$ + ")"
grpErrors.TickEvery = grpErrors.NumPoints / 5
grpErrors.LabelEvery = grpErrors.TickEvery
For i = 1 To nCount
  grpErrors.GraphData = ErrorBuffer(i)
Next
grpLinks.Refresh
grpErrors.DrawMode = 3
```

```
    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  View Links Data
    grpLinks.DataReset = 9
    grpLinks.NumPoints = nWts
    b$ = "Last Loaded Weights after" + Str$(nEpochCount) + " Epochs: "
    For i = 0 To nWts - 1
       grpLinks.GraphData = Weights(i)
       b$ = b$ + Str$(Weights(i)) + " / "
    Next
    Text.Text = Text.Text + CrLf$ + b$
    grpLinks.GraphTitle = "Weights (" + LinksFileName$ + ") after" + Str$(nEpochCount) + " epochs"
    grpLinks.TickEvery = nNeurons(1)
    grpLinks.LabelEvery = grpLinks.TickEvery
    grpLinks.Refresh
    grpLinks.DrawMode = 3

    GotLinksFile = True
    HaveTrainedNet = False                  ' Weights have been redefined

End Sub

-----------------------------------------------------------------------------------------------------------------


Sub ReadTrainingSet (nTRN, nPatterns, Xmapped(), Ymapped())
Dim i, J, x, Y, a$, b$, KeyWord$

   Do
      dummy = DoEvents()
      If EOF(nTRN) Then Exit Do
      KeyWord$ = InputField$("[]", nTRN)
      Select Case KeyWord$
      Case "INPUT RANGES"                  ' nInputs lines
      i = 0                                '    end this group with a blank line
         Do                                '    (number of nInputs determined by
            Line Input #nTRN, a$           '      the number of lines that follow)
            If Trim$(a$) = "" Then Exit Do '    InputNeuronName, lo, hi
            i = i + 1                      '    ...
            InputName$(i) = Trim$(Parm$(a$, t$))
            rMinInput(i) = Val(Parm$(a$, t$))
            rMaxInput(i) = Val(a$)
         Loop
         nInputs = i:          txtNinputs.Text = nInputs
         nNeurons(0) = nInputs

      Case "OUTPUT RANGES"                 ' nOutputs lines
         i = 0                             '    end this group with a blank line
         Do                                '
            Line Input #nTRN, a$           '    OutputNeuronName, lo, hi
            If Trim$(a$) = "" Then Exit Do '    ...
            i = i + 1
            OutputName$(i) = Trim$(Parm$(a$, t$))
```

```
            rMinOutput(i) = Val(Parm$(a$, t$))
            rMaxOutput(i) = Val(a$)
         Loop
         nOutputs = i:          txtNoutputs.Text = nOutputs

         ' Hidden Layers from screen input
         a$ = txtNhidden.Text
         b$ = ""
         For i = 1 To MaxLayers - 1
            nNeurons(i) = Val(Parm$(a$, ","))
            b$ = b$ + Str$(nNeurons(i)) + ","
            If a$ = "" Then Exit For
         Next:                  txtNhidden.Text = Left$(Trim$(b$), Len(b$) - 2)
         nHiddenLayers = i:
         nLayers = nHiddenLayers + 2
         LastHiddenLayer = i
         LastLayer = i + 1
         nNeurons(LastLayer) = nOutputs

      Case "TRAINING DATA"              ' (nInputs + nOutputs) pieces of data
         J = 0                          '    program reads to the end of file
         Do                             '    counter gives number of training patterns
            dummy = DoEvents()

            If EOF(nTRN) Then Exit Do
            Line Input #nTRN, a$
            If Trim$(a$) <> "" Then
               J = J + 1
               For i = 1 To nInputs     ' Mapping input vector to to xMapped(i,j) and yMapped(i,j)
                  x = Val(Parm$(a$, t$))
                  Xmapped(i, J) = Map(x, rMinInput(i), rMaxInput(i), ScaleInLo, ScaleInHi)
               Next
               For i = 1 To nOutputs
                  Y = Val(Parm$(a$, t$))
                  Ymapped(i, J) = Map(Y, rMinOutput(i), rMaxOutput(i), ScaleOutLo, ScaleOutHi)
               Next
            End If
         Loop

         nPatterns = J:         txtNpatterns.Text = Str$(nPatterns)
         Exit Do
      End Select
   Loop
   Close #nTRN

End Sub
```

```
Sub RunNet (VectorIn(), rMinInput(), rMaxInput(), VectorOut(), rMinOutput(), rMaxOutput())
ReDim rNetInput(MaxNeuronsPerLayer), rNetOutput(MaxNeuronsPerLayer)
Dim i, n

   For i = 1 To nInputs
      rNetInput(i) = Map(VectorIn(i), rMinInput(i), rMaxInput(i), ScaleInLo, ScaleInHi)
   Next

   iCode = PutSlab(NetNumber, 0, rNetInput(1)):    If iCode GoTo ErrorRunNet

   For n = 1 To LastLayer
      iCode = BpPropagate(NetNumber, n):    If iCode GoTo ErrorRunNet
   Next

   iCode = GetSlab(NetNumber, LastLayer, rNetOutput(1)):    If iCode GoTo ErrorRunNet

   For i = 1 To nOutputs
      VectorOut(i) = Map(rNetOutput(i), ScaleOutLo, ScaleOutHi, rMinOutput(i), rMaxOutput(i))
   Next

Exit Sub

ErrorRunNet:
   MsgBox NetError$(iCode), 16, "Error in RunNet"

End Sub
```

```
Sub RunSimulator ()
ReDim VectorIn(MaxNeuronsPerLayer), VectorOut(MaxNeuronsPerLayer)
Dim i, iter, a$, b$, nSIM
Dim SimulationFileName$
Dim Tmid, TmidOld, Ttop, HRtp, StPt, StPtOld, StPtAdj, ActualStPtAdj, StPtMin, StPtMax

   SimulationFileName$ = NewName$(LinksFileName$, "SIM")
   diaFiles.Filename = SimulationFileName$
   OpenFile "Simulation Run File", "sim", "Output", SimulationFileName$, nSIM
   lblSIM.Caption = SimulationFileName$
   If SimulationFileName$ = "" Then lblSIM.Caption = "Simulation Run Not Saved"

   Print #nSIM, "Links from: "; LinksFileName$; "    After "; nEpochCount; " Epochs"
   Print #nSIM, " "
   Print #nSIM, "time"; t$; "Tmid"; t$; "Ttop"; t$; "HRtp"; t$; "Taut"
   Print #nSIM, "0"; t$; "80"; t$; "80"; t$; "0"; t$; "80"

   ' Get NN input state vector from Simulator on COMM port
   ' RunNet and send resulting SetPointAdjustment to Simulator
```

```
' initialize
iter = 0
StPtOld = 80
StPtAdj = 0

StPtMin = 80
StPtMax = 450

' Reset SIM
COMM.Output = "99" + Chr$(13)

frmSIM.Show
frmSIM.Caption = "Simulated Autoclave"
frmSIM.grpTemp.GraphTitle = "T, " + LinksFileName$ + ", after" + Str$(nEpochCount) + " epochs"

frmSIM.grpTemp.NumSets = 2
frmSIM.grpTemp.AutoInc = False

' Do cycles
iter = 0
Do While iter < 400
   iter = iter + 1
   dummy = DoEvents()

   ' Get SIM data from COMM port
   a$ = ""
   Do
      a$ = a$ + COMM.Input
      dummy = DoEvents()
   Loop Until InStr(a$, Chr$(13))
   lblGeneral.Caption = a$

   b$ = "Data From Simulator:"
   For i = 1 To nInputs
      VectorIn(i) = Val(Parm$(a$, ","))
      b$ = b$ + CrLf$ + InputName$(i) + ": " + Format$(VectorIn(i), " ###0.00")
   Next

   Tmid = VectorIn(1)
   Ttop = VectorIn(1) - VectorIn(2)
   HRtp = VectorIn(5)

   ' Run net and send resulting set point adjustment to SIM
   RunNet VectorIn(), rMinInput(), rMaxInput(), VectorOut(), rMinOutput(), rMaxOutput()
   StPtAdj = VectorOut(1)
   COMM.Output = Str$(StPtAdj) + Chr$(13)

   ' SIM will clip StPt
   StPt = Between((StPtOld + StPtAdj), StPtMin, StPtMax)
   ActualStPtAdj = StPt - StPtOld
```

```
        b$ = b$ + CrLf$ + "StPtAdj by NN: " + Format$(StPtAdj, " ###0.0")
        b$ = b$ + CrLf$ + "Actual StPtAdj: " + Format$(ActualStPtAdj, " ###0.0")
        b$ = b$ + CrLf$ + "Set Point: " + Format$(StPt, " ###0.0")
        Text.Text = b$

        dummy = DoEvents()

        frmSIM.grpTemp.YAxisMax = 600
        frmSIM.grpTemp.NumPoints = (iter \ 25 + 1) * 25
        frmSIM.grpTemp.TickEvery = frmSIM.grpTemp.NumPoints / 5
        frmSIM.grpTemp.LabelEvery = frmSIM.grpTemp.TickEvery
        frmSIM.grpTemp.ThisSet = 1
        frmSIM.grpTemp.ThisPoint = iter
        frmSIM.grpTemp.GraphData = StPt
        frmSIM.grpTemp.ThisSet = 2
        frmSIM.grpTemp.GraphData = Tmid
        frmSIM.grpTemp.Refresh
        frmSIM.grpTemp.DrawMode = 3

        dummy = DoEvents()

        TmidOld = Tmid
        StPtOld = StPt

        If (iter Mod 4) = 0 Then
           Print #nSIM, iter; t$; Tmid; t$; Ttop; t$; HRtp; t$; StPt
        End If
    Loop

    Close #nSIM

    frmSIM.grpTemp.GraphType = 6
    frmSIM.grpTemp.GraphStyle = 0
    frmSIM.grpTemp.PrintStyle = 1
    frmSIM.grpTemp.DrawMode = 3

    MsgBox "Simulation Run Completed.", 64, ""

End Sub

-----------------------------------------------------------------------------------------------------------------------------------------------------------


Sub RunTestFile ()
ReDim ExpectedOut(MaxNeuronsPerLayer), dev(MaxNeuronsPerLayer), devR(MaxNeuronsPerLayer)
ReDim Emin(MaxNeuronsPerLayer), Emax(MaxNeuronsPerLayer), Esum(MaxNeuronsPerLayer)
ReDim VectorIn(MaxNeuronsPerLayer), VectorOut(MaxNeuronsPerLayer)
Dim i, nTST, nTSO, nCount
Dim a$, b0$, b1$, b2$, b3$, b4$, TestFileName$, TestOutputFileName$

    OpenFile "Test Data File", "tst", "Input", TestFileName$, nTST
```

```
If TestFileName$ = "" Then Exit Sub

TestOutputFileName$ = NewName$(LinksFileName$, "TSO")
diaFiles.Filename = TestOutputFileName$
OpenFile "Test Output File", "tso", "Output", TestOutputFileName$, nTSO
If TestOutputFileName$ = "" Then Exit Sub

lblMiscFile.Caption = "Write to: " + TestOutputFileName$
Print #nTSO, "Trained by: "; lblTRN.Caption
Print #nTSO, "Links from: "; LinksFileName$; "     After "; nEpochCount; " Epochs"
Print #nTSO, "The weights are:"
For i = 0 To nTotalWeights(nLayers, nNeurons()) - 1
   Print #nTSO, Weights(i); t$;
Next:    Print #nTSO,
Print #nTSO,
Print #nTSO, "The results are:"
For i = 1 To nOutputs
   Print #nTSO, "Expected("; Trim$(Str$(i)); ")"; t$;
   Print #nTSO, "NNcalc'd("; Trim$(Str$(i)); ")"; t$;
   Print #nTSO, "RelErr("; Trim$(Str$(i)); ")"; t$;
Next:    Print #nTSO,

For i = 1 To nOutputs
   Emax(i) = -1E+08!
   Emin(i) = 1E+08!
   Esum(i) = 0!
Next

nCount = 0
Do Until EOF(nTST)
   nCount = nCount + 1
   dummy = DoEvents()
   Line Input #nTST, a$
   For i = 1 To nInputs
      VectorIn(i) = Val(Parm$(a$, t$))
   Next
   For i = 1 To nOutputs
      ExpectedOut(i) = Val(Parm$(a$, t$))
   Next
   RunNet VectorIn(), rMinInput(), rMaxInput(), VectorOut(), rMinOutput(), rMaxOutput()
   For i = 1 To nOutputs
      dev(i) = (VectorOut(i) - ExpectedOut(i))
      devR(i) = dev(i) / ExpectedOut(i)
      If devR(i) > Emax(i) Then Emax(i) = devR(i)
      If devR(i) < Emin(i) Then Emin(i) = devR(i)
      Esum(i) = Esum(i) + dev(i) ^ 2
      Print #nTSO, ExpectedOut(i); t$; VectorOut(i); t$; devR(i); t$;
   Next
   Print #nTSO,
Loop
Print #nTSO,
```

```
    b0$ = "Number of test vectors: " + Str$(nCount)
    Text.Text = b0$
    Print #nTSO, b0$
    For i = 1 To nOutputs
       b1$ = "For Output" + Str$(i) + ":     "
       b2$ = "MinDev = " + Format$(Emin(i), "00.00%") + "     "
       b3$ = "MaxDev = " + Format$(Emax(i), "00.00%") + "     "
       b4$ = "E^2/N = " + Format$(Esum(i) ^ 2 / nCount, "0.000E+00")
       Print #nTSO, b1$ + b2$ + b3$ + b4$
       b0$ = b0$ + CrLf$ + b1$ + CrLf$ + b2$ + b3$ + b4$
    Next
    Text.Text = b0$

    Close #nTST
    Close #nTSO

    btnRunTestFile.Value = False
    MsgBox "RunTestFile Completed.", 64, ""

End Sub

-------------------------------------------------------------------------------------------------------------------------------


Function SmallerOf (a As Single, b As Single)
    If a < b Then
       SmallerOf = a
    Else
       SmallerOf = b
    End If
End Function

-------------------------------------------------------------------------------------------------------------------------------


Static Sub TrainNet ()
ReDim Index(MaxPatterns)
ReDim Xmapped(MaxNeuronsPerLayer, MaxPatterns), Ymapped(MaxNeuronsPerLayer, MaxPatterns)
ReDim rNetInput2(MaxNeuronsPerLayer, MaxPatterns), rNetOutput2(MaxNeuronsPerLayer,
MaxPatterns)
Dim nLoop As Long, LastSave As Long
Dim b$, RNDflag$, TrainingSetFileName$, LNKdescription$
Dim i, J, n, nTRN, nLNK, nPatterns
Dim SumError, Emin, Emax, ErrorFactor, AvgError

    If TrainMore = 0 Or TrainMore = 1 Then
       OpenFile "Training Set File", "trn", "Input", TrainingSetFileName$, nTRN
       lblTRN.Caption = TrainingSetFileName$
       If TrainingSetFileName$ = "" Then Exit Sub
```

```
    ReadTrainingSet nTRN, nPatterns, Xmapped(), Ymapped()

    LNKdescription$ = lblLNK.Caption
    OpenFile "Links File", "lnk", "Output", LinksFileName$, nLNK    ' See ReadLinksFile
    lblLNK.Caption = LinksFileName$
    If LinksFileName$ = "" Then Exit Sub

    PrintLinksFileHeader nLNK, nPatterns, LNKdescription$

    CreateNet
    txtTRN.Text = "Train Net" + Str$(NetNumber)
    If TrainMore = 1 Then
       PutLinkWeights
    End If

  Else

    nTRN = FreeFile
    Open lblTRN.Caption For Input As nTRN

    ReadTrainingSet nTRN, nPatterns, Xmapped(), Ymapped()

    btnLNK.Value = True
    nLNK = FreeFile
    b$ = lblLNK.Caption
    Open Parm$(b$, ",") For Append As nLNK

  End If


  If TrainMore = 0 Then
    nLoop = 0
    grpErrors.DataReset = 9
    grpErrors.GraphData = 0
  Else
    nLoop = LastEpochCount
    If nLoop <= nEpochCount Then
       nEpochCount = nEpochCount * 2
       txtEpochCount.Text = Trim$(Str$(nEpochCount))
    End If
    grpErrors.DataReset = 9
    grpErrors.NumPoints = ((LastEpochCount / nSkip) \ 25 + 1) * 25
    grpErrors.GraphData = ErrorBuffer(1)
    For i = 1 To LastEpochCount / nSkip
       grpErrors.GraphData = ErrorBuffer(i)
    Next
  End If
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''' Training

Do Until nLoop >= nEpochCount          ' nEpochCount is defined through txtEpochCount.Text
   dummy = DoEvents()

   nLoop = nLoop + 1:    txtEpochs.Text = nLoop

   SumError = 0!
   Emin = 1E+08!
   Emax = -1E+08!

   For i = 1 To nPatterns                ' Generate Index Vector
      Index(i) = I                        '       1, 2, 3, ... nPatterns
   Next i
   If chkRandomIndex.Value Then
      RNDflag$ = "Randomized"
      RandomIndex nPatterns, Index()    ' Randomize indices in Index()
   Else
      RNDflag$ = ""
   End If

   For J = 1 To nPatterns                ' Prep data arrays rNetInput2(i,j)
      For i = 1 To nInputs               '                  and rNetOutput2(i,j)
         rNetInput2(i, Index(J)) = Xmapped(i, J)
      Next
      For i = 1 To nOutputs
         rNetOutput2(i, Index(J)) = Ymapped(i, J)
      Next
   Next

   For J = 1 To nPatterns
      dummy = DoEvents()

      ' Put input pattern
      iCode = PutSlab(NetNumber, 0, rNetInput2(1, J)):   If iCode Then GoTo ErrorTrainNet

      ' Propagate through hidden layers
      For n = 1 To LastLayer
         iCode = BpPropagate(NetNumber, n):   If iCode Then GoTo ErrorTrainNet
      Next

      ' Evaluate error
      iCode = BpEvaluate(NetNumber, LastLayer, rNetOutput2(1, J), ErrorFactor)
         If iCode Then GoTo ErrorTrainNet

      ' Back Propagate
      For n = LastHiddenLayer To 0 Step -1
         iCode = BpTrain(NetNumber, n, rLearningRate, rMomentum, 0)
            If iCode Then GoTo ErrorTrainNet
      Next
```

```
        SumError = SumError + ErrorFactor
        If ErrorFactor > Emax Then Emax = ErrorFactor
        If ErrorFactor < Emin Then Emin = ErrorFactor

    Next

    AvgError = SumError / nPatterns
    lblGeneral.Caption = "Avg. Error:    " + Str$(AvgError)

    If (nLoop Mod nSkip) = 0 Then
       grpErrors.NumPoints = ((nLoop / nSkip) \ 25 + 1) * 25
       grpErrors.TickEvery = grpErrors.NumPoints / 5
       grpErrors.LabelEvery = grpErrors.TickEvery
       grpErrors.ThisPoint = nLoop / nSkip + 1
       grpErrors.GraphData = AvgError
       grpErrors.Refresh
       grpErrors.DrawMode = 3

       ' Get link weights of net in a vector
       GetLinkWeights
       LastSave = nLoop

       ' Store weights in LNK file
       b$ = "nLoop=" + Str$(nLoop) + ": "
       Print #nLNK, nLoop;
       For i = 0 To nTotalWeights(nLayers, nNeurons()) - 1
          Print #nLNK, t$; Weights(i);
          b$ = b$ + Str$(Weights(i)) + " / "
       Next
       If (chkDisplay.Value) Then Text.Text = b$
       Print #nLNK, t$; AvgError; t$; Emin; t$; Emax; t$; RNDflag$

    End If

Loop
nEpochCount = LastSave
LastEpochCount = nEpochCount
Close #nLNK

' Display last saved net
grpLinks.DataReset = 9
grpLinks.NumPoints = nTotalWeights(nLayers, nNeurons())
For i = 0 To nTotalWeights(nLayers, nNeurons())
   grpLinks.GraphData = Weights(i)
Next
grpLinks.TickEvery = nNeurons(1)
grpLinks.LabelEvery = grpLinks.TickEvery
grpLinks.Refresh
grpLinks.DrawMode = 3

txtTRN.ForeColor = GRAY
```

```
    HaveTrainedNet = True
    TrainMore = 0

    MsgBox "Training is Complete.", 64, ""

Exit Sub

ErrorTrainNet:
    MsgBox NetError$(iCode), 16, "Error in TrainNet"

End Sub
```

```
Sub txtCOMMport_DblClick ()
    If txtCOMMport.Text = 2 Then
        txtCOMMport.Text = 1
        COMM.CommPort = 1
    Else
        txtCOMMport.Text = 2
        COMM.CommPort = 2
    End If
End Sub
```

```
Sub txtEpochCount_DblClick ()
    nEpochCount = Val(txtEpochCount.Text)
    txtEpochCount.ForeColor = RED
End Sub
```

```
Sub txtEpochCount_LostFocus ()
    txtEpochCount.ForeColor = BLACK
End Sub
```

```
Sub txtNhidden_Change ()
    txtNhidden.ForeColor = &H80000008
End Sub
```

```
Sub txtNrecords_Change ()
    nSkip = txtNrecords.Text
    grpErrors.BottomTitle = "Epochs / " + txtNrecords.Text
End Sub
```

APPENDIX B

Thin Panel (32 plies) Training Set
Thick Panel (256 plies) Training Set

Thin Panel (32 plies) Training Set

[Input Ranges]

Tmid	80	500
Tmid-Ttop	-10	50
Tmid-Taut	-100	200
dT/dt	-2.5	5
Hr	0	4

[Output Ranges]

Tadj	-12	12

[Training Data]

Tmid (Y2)	id-Ttop (	id-Taut (	dT/dt (Y1)	Hr (Y1)	Tadj (Y2)	not used	not used
80.03	-0.03	-0.97	0.03	0	1	0	5.96E-02
80.13	-0.08	-1.87	0.10	0	1	0	0.141795
80.30	-0.12	-2.70	0.17	0	1	0	0.212571
80.54	-0.15	-3.46	0.24	0	1	0	0.27692
80.84	-0.19	-4.16	0.30	0	1	0	0.335851
81.22	-0.25	-5.78	0.38	0	2	0	0.44768
81.73	-0.33	-7.27	0.51	0	2	0	0.579637
82.35	-0.40	-8.65	0.63	0	2	0	0.696206
83.09	-0.46	-9.91	0.74	0	2	0	0.802616
83.94	-0.52	-11.06	0.84	0	2	0	0.900189
84.90	-0.60	-13.10	0.96	0	3	0	1.047527
86.02	-0.70	-14.98	1.12	0	3	0	1.212119
87.29	-0.78	-16.71	1.27	0	3	0	1.358687
88.70	-0.86	-18.30	1.41	0	3	0	1.492678
90.24	-0.94	-19.76	1.54	0	3	0	1.615615
91.93	-1.04	-22.07	1.69	0	4	0	1.786283
93.80	-1.14	-24.20	1.87	0	4	0	1.97235
95.84	-1.24	-26.16	2.04	0	4	0	2.138695
98.04	-1.33	-27.96	2.20	0	4	0	2.290918
100.41	-1.45	-30.59	2.37	0	5	0	2.488431
102.99	-1.56	-33.01	2.58	0	5	0	2.699205
105.77	-1.68	-35.23	2.78	0	5	0	2.888351
108.73	-1.78	-37.27	2.96	0	5	0	3.061662
111.88	-1.90	-40.12	3.15	0	6	0	3.278737
115.26	-2.03	-42.74	3.38	0	6	0	3.507734
118.85	-2.15	-45.15	3.59	0	6	0	3.713952
122.64	-2.26	-47.36	3.79	0	6	0	3.903373
126.65	-2.40	-50.35	4.00	0	7	0	4.1358
130.89	-2.53	-53.11	4.24	0	7	0	4.37965
135.37	-2.66	-55.63	4.48	0	7	0	4.600505
140.06	-2.77	-57.94	4.69	0	7	0	4.804676

144.97	-2.91	-61.03	4.92	0	8	0	5.052374
150.15	-3.05	-63.85	5.17	2.20E-02	8	0	5.312606
155.57	-3.18	-66.43	5.43	6.43E-02	8	0	5.551676
161.24	-3.29	-68.76	5.66	0.117212	8	0	5.776798
167.15	-3.42	-71.85	5.92	0.156627	9	0	6.0494
173.36	-3.55	-74.64	6.21	0.238865	9	0	6.340301
179.86	-3.67	-77.14	6.50	0.353867	9	0	6.618149
186.66	-3.77	-79.34	6.80	0.504955	9	0	6.893203
193.53	-3.55	-72.47	6.87	0.949279	10	53.10401	6.651446
200.28	-3.51	-74.72	6.75	0.980708	9	111.9465	6.711061
207.30	-3.53	-75.70	7.03	1.25982	8	187.5357	7.05021
214.66	-3.53	-76.34	7.35	1.662427	8	287.2813	7.347373
222.38	-3.49	-76.62	7.72	2.180229	8	418.095	7.685456
230.51	-3.38	-75.49	8.13	2.877628	7	590.7527	8.019688
239.06	-3.22	-73.94	8.56	3.738132	7	815.0406	8.396213
248.14	-3.01	-71.86	9.08	4.817247	7	1104.075	8.864664
257.83	-2.70	-68.18	9.68	6.185645	6	1475.214	9.372884
268.16	-2.31	-63.84	10.34	7.799403	6	1943.178	9.947742
279.21	-1.86	-58.79	11.05	9.634493	6	2521.248	10.59763
290.93	-1.33	-52.07	11.72	11.59281	5	3216.816	11.19088
303.11	-0.78	-44.89	12.17	13.33299	5	4016.796	11.62213
315.33	-0.29	-37.67	12.22	14.46556	5	4884.729	11.74073
327.02	0.08	-29.98	11.69	14.65614	4	5764.098	11.3125
337.55	0.29	-23.45	10.53	13.74067	4	6588.538	10.31927
346.83	0.40	-18.17	9.28	12.49955	4	7338.511	9.173539
356.64	0.91	-11.36	9.81	14.04246	3	8181.059	9.304344
367.04	1.50	-3.96	10.40	15.97306	3	9139.442	9.807139
378.01	2.13	4.01	10.97	17.99305	3	10219.02	10.34449
389.53	2.78	12.53	11.52	20.0476	3	11421.88	10.86158
401.47	3.44	21.47	11.94	21.97001	3	12740.08	11.28354
413.57	4.05	30.57	12.10	23.48679	3	14149.29	11.48783
425.38	4.56	39.38	11.81	24.22524	3	15602.8	11.30615
436.26	4.88	47.26	10.88	23.79906	3	17030.75	10.56142
445.50	4.96	53.50	9.24	21.99721	3	18350.58	9.155375
452.52	4.80	57.52	7.02	18.98464	3	19489.66	7.173725
457.09	4.47	59.09	4.58	15.311	3	20408.32	4.910107
459.42	4.05	58.42	2.32	11.65052	3	21107.35	2.745095
459.94	3.61	55.94	0.53	8.491117	3	21616.82	0.968264
459.22	3.19	52.22	-0.73	6.02085	3	21978.07	-0.30846
457.71	2.80	47.71	-1.50	4.207082	3	22230.49	-1.11694
455.80	2.45	42.80	-1.91	2.92238	3	22405.84	-1.55491
453.75	2.12	37.75	-2.05	2.028176	3	22527.53	-1.72787
451.73	1.82	32.73	-2.02	1.409278	3	22612.08	-1.72333
449.84	1.54	27.84	-1.88	0.980321	3	22670.9	-1.60572
448.14	1.32	24.14	-1.70	0.708764	2	22713.43	-1.47743
446.61	1.12	20.61	-1.53	0.502471	2	22743.58	-1.33526
445.24	0.97	18.24	-1.37	0.378731	1	22766.3	-1.2205
444.00	0.85	16.00	-1.24	0.274114	1	22782.75	-1.11749

442.89	0.74	13.89	-1.11	0.193702	1	22794.37	-0.99931
441.88	0.67	12.88	-1.01	0.162308	0	22804.11	-0.93425
440.93	0.61	11.93	-0.95	0.123208	0	22811.5	-0.89372
440.04	0.56	11.04	-0.89	8.94E-02	0	22816.87	-0.84453
439.20	0.52	10.20	-0.84	6.41E-02	0	22820.71	-0.79324
438.42	0.48	9.42	-0.78	4.51E-02	0	22823.42	-0.74188
437.69	0.44	8.69	-0.73	3.10E-02	0	22825.28	-0.6916
437.01	0.41	8.01	-0.68	2.05E-02	0	22826.51	-0.64308
436.39	0.37	7.39	-0.63	1.27E-02	0	22827.27	-0.59676
435.80	0.34	6.80	-0.58	6.91E-03	0	22827.68	-0.55286
435.26	0.32	6.26	-0.54	2.68E-03	0	22827.84	-0.51152
434.77	0.29	5.77	-0.50	0	0	22827.82	-0.47275
434.31	0.27	5.31	-0.46	0	0	22827.66	-0.43652
433.88	0.25	4.88	-0.42	0	0	22827.42	-0.40277
433.49	0.23	4.49	-0.39	0	0	22827.11	-0.3714
433.13	0.21	4.13	-0.36	0	0	22826.76	-0.34229
432.80	0.19	3.80	-0.33	0	0	22826.39	-0.31532
432.49	0.18	3.49	-0.31	0	0	22826.01	-0.29037
432.21	0.16	3.21	-0.28	0	0	22825.63	-0.26731
431.95	0.15	2.95	-0.26	0	0	22825.26	-0.24601
431.71	0.14	2.71	-0.24	0	0	22824.89	-0.22635
431.49	0.13	2.49	-0.22	0	0	22824.54	-0.20823
431.29	0.12	2.29	-0.20	0	0	22824.2	-0.19152
431.11	0.11	2.11	-0.19	0	0	22823.88	-0.17613
430.91	0.13	2.91	-0.20	2.24E-02	-1	22825.23	-0.2197
430.63	0.20	4.63	-0.28	0.053943	-2	22828.46	-0.34669
430.22	0.28	6.22	-0.41	5.59E-02	-2	22831.82	-0.4879
429.68	0.35	7.68	-0.54	5.16E-02	-2	22834.91	-0.61224
429.00	0.45	10.00	-0.68	7.47E-02	-3	22839.4	-0.78332
428.13	0.55	12.13	-0.86	7.48E-02	-3	22843.88	-0.96942
427.10	0.65	14.10	-1.04	6.89E-02	-3	22848.02	-1.13563
425.90	0.74	15.90	-1.20	6.33E-02	-3	22851.82	-1.28758
424.53	0.86	18.53	-1.37	8.55E-02	-4	22856.95	-1.4847
422.93	1.01	21.93	-1.60	0.112053	-5	22863.67	-1.7527
421.06	1.17	25.06	-1.87	0.109375	-5	22870.23	-2.02343
418.94	1.31	27.94	-2.12	0.100782	-5	22876.28	-2.26679
416.58	1.44	30.58	-2.36	9.26E-02	-5	22881.84	-2.48945
413.99	1.60	33.99	-2.60	0.112419	-6	22888.58	-2.75148
411.10	1.79	38.10	-2.89	0.136788	-7	22896.79	-3.07909
407.85	2.01	42.85	-3.24	0.159482	-8	22906.36	-3.46234
404.23	2.22	47.23	-3.62	0.152968	-8	22915.54	-3.83828
400.26	2.43	51.26	-3.98	0.140839	-8	22923.99	-4.17809
395.95	2.61	54.95	-4.30	0.129391	-8	22931.75	-4.4893
391.35	2.78	58.35	-4.60	0.118866	-8	22938.88	-4.7749
386.47	2.94	61.47	-4.88	0.109199	-8	22945.43	-5.03717
381.34	3.08	64.34	-5.13	0.100319	-8	22951.45	-5.27808
375.97	3.22	66.97	-5.37	9.22E-02	-8	22956.98	-5.49939
370.39	3.34	69.39	-5.58	8.47E-02	-8	22962.06	-5.70269

364.61	3.45	71.61	-5.78	7.78E-02	-8	22966.73	-5.88947
358.65	3.55	73.65	-5.96	7.15E-02	-8	22971.02	-6.06106
352.53	3.65	75.53	-6.12	6.56E-02	-8	22974.96	-6.2187
346.25	3.73	77.25	-6.28	6.03E-02	-8	22978.57	-6.36352
339.83	3.81	78.83	-6.42	5.54E-02	-8	22981.9	-6.49657
333.29	3.89	80.29	-6.55	5.09E-02	-8	22984.95	-6.6188
326.62	3.95	81.62	-6.66	0.046764	-8	22987.76	-6.73109
319.85	4.02	82.85	-6.77	4.30E-02	-8	22990.34	-6.83426
312.98	4.07	83.98	-6.87	3.95E-02	-8	22992.71	-6.92903
306.02	4.13	85.02	-6.96	3.63E-02	-8	22994.88	-7.0161
298.97	4.17	85.97	-7.05	3.33E-02	-8	22996.88	-7.0961
291.84	4.22	86.84	-7.13	3.06E-02	-8	22998.71	-7.16958
284.65	4.26	87.65	-7.20	2.81E-02	-8	23000.4	-7.2371
277.38	4.29	88.38	-7.26	2.58E-02	-8	23001.95	-7.29912
270.06	4.33	89.06	-7.32	2.37E-02	-8	23003.38	-7.35611
262.68	4.36	89.68	-7.38	2.18E-02	-8	23004.68	-7.40846
255.26	4.39	90.26	-7.43	2.00E-02	-8	23005.88	-7.45655
247.78	4.42	90.78	-7.47	1.84E-02	-8	23006.99	-7.50073
240.27	4.44	91.27	-7.52	1.69E-02	-8	23008	-7.54132
232.71	4.46	91.71	-7.56	1.55E-02	-8	23008.94	-7.57861
225.12	4.48	92.12	-7.59	1.43E-02	-8	23009.79	-7.61287
217.49	4.50	92.49	-7.63	0.013107	-8	23010.58	-7.64435
209.84	4.52	92.84	-7.66	0.012041	-8	23011.3	-7.67326
202.15	4.53	93.15	-7.68	1.11E-02	-8	23011.96	-7.69983
194.44	4.55	93.44	-7.71	1.02E-02	-8	23012.57	-7.72423
186.71	4.56	93.71	-7.73	9.34E-03	-8	23013.13	-7.74665
178.93	4.61	94.93	-7.78	3.59E-02	-9	23015.29	-7.825
171.11	4.53	91.11	-7.82	0	-10	23009.43	-7.74469
163.72	4.20	83.72	-7.39	0	-11	22994.58	-7.063
156.91	3.87	76.91	-6.81	0	-12	22980.32	-6.4715
150.66	3.56	70.66	-6.25	0	-11	22967.19	-5.94087
144.91	3.27	64.91	-5.74	0	-10	22955.12	-5.45628
139.63	3.00	59.63	-5.28	0	-9	22944.04	-5.0121
134.79	2.76	54.79	-4.85	0	-8	22933.86	-4.60439
130.33	2.53	50.33	-4.45	0	-7	22924.51	-4.22996
126.24	2.33	46.24	-4.09	0	-6	22915.91	-3.88603
122.48	2.14	42.48	-3.76	0	-5	22908.02	-3.57008
119.03	1.96	39.03	-3.45	0	-4	22900.77	-3.27983
115.85	1.81	35.85	-3.17	0	-3	22894.11	-3.01317
112.94	1.66	32.94	-2.91	0	-2	22887.98	-2.7682
110.26	1.52	30.26	-2.68	0	-2	22882.36	-2.54314
107.80	1.40	27.80	-2.46	0	-2	22877.2	-2.33638
105.54	1.29	25.54	-2.26	0	-2	22872.45	-2.14643
103.46	1.18	23.46	-2.08	0	-2	22868.09	-1.97192
101.56	1.09	21.56	-1.91	0	-2	22864.08	-1.8116
99.80	1.00	19.80	-1.75	0	-2	22860.4	-1.66432
98.19	0.92	18.19	-1.61	0	-2	22857.02	-1.52901
96.71	0.84	16.71	-1.48	0	-2	22853.92	-1.4047

95.36	0.77	15.36	-1.36	0	-2	22851.06	-1.29049
94.11	0.71	14.11	-1.25	0	-2	22848.44	-1.18557
92.96	0.65	12.96	-1.15	0	-2	22846.04	-1.08919
91.91	0.60	11.91	-1.05	0	-2	22843.82	-1.00063
90.94	0.55	10.94	-0.97	0	-2	22841.79	-0.91928
90.05	0.51	10.05	-0.89	0	-1	22839.92	-0.84454
89.23	0.46	9.23	-0.82	0	0	22838.21	-0.77588
88.48	0.43	8.48	-0.75	0	0	22836.63	-0.7128
87.79	0.39	7.79	-0.69	0	0	22835.18	-0.65485
87.16	0.36	7.16	-0.63	0	0	22833.85	-0.60161
86.58	0.33	6.58	-0.58	0	0	22832.63	-0.5527
86.04	0.30	6.04	-0.53	0	0	22831.51	-0.50776
85.55	0.28	5.55	-0.49	0	0	22830.48	-0.46648
85.10	0.26	5.10	-0.45	0	0	22829.53	-0.42856
84.68	0.24	4.68	-0.41	0	0	22828.66	-0.39371
84.30	0.22	4.30	-0.38	0	0	22827.86	-0.3617
83.95	0.20	3.95	-0.35	0	0	22827.12	-0.3323
83.63	0.18	3.63	-0.32	0	0	22826.45	-0.30528
83.34	0.17	3.34	-0.30	0	0	22825.83	-0.28046
83.07	0.15	3.07	-0.27	0	0	22825.26	-0.25766
82.82	0.14	2.82	-0.25	0	0	22824.73	-0.23671
82.59	0.13	2.59	-0.23	0	0	22824.25	-0.21747
82.38	0.12	2.38	-0.21	0	0	22823.81	-0.19979
82.18	0.11	2.18	-0.19	0	0	22823.41	-0.18354
82.01	0.10	2.01	-0.18	0	0	22823.03	-0.16862
81.84	0.09	1.84	-0.16	0	0	22822.69	-0.15491
81.69	0.09	1.69	-0.15	0	0	22822.38	-0.14232
81.56	0.08	1.56	-0.14	0	0	22822.09	-0.13075
81.43	0.07	1.43	-0.13	0	0	22821.82	-0.12012
81.31	0.07	1.31	-0.12	0	0	22821.58	-0.11035
81.21	0.06	1.21	-0.11	0	0	22821.35	-0.10138
81.11	0.06	1.11	-0.10	0	0	22821.15	-9.31E-02
81.02	0.05	1.02	-0.09	0	0	22820.96	-8.56E-02
80.94	0.05	0.94	-0.08	0	0	22820.79	-7.86E-02
80.86	0.04	0.86	-0.08	0	0	22820.63	-7.22E-02
80.79	0.04	0.79	-0.07	0	0	22820.48	-6.63E-02
80.73	0.04	0.73	-0.06	0	0	22820.34	-6.10E-02
80.67	0.03	0.67	-0.06	0	0	22820.22	-5.60E-02
80.61	0.03	0.61	-0.05	0	0	22820.11	-5.14E-02
80.56	0.03	0.56	-0.05	0	0	22820	-4.73E-02
80.52	0.03	0.52	-0.05	0	0	22819.91	-4.34E-02
80.47	0.02	0.47	-0.04	0	0	22819.82	-3.99E-02
80.44	0.02	0.44	-0.04	0	0	22819.74	-3.66E-02
80.40	0.02	0.40	-0.04	0	0	22819.66	-3.37E-02
80.37	0.02	0.37	-0.03	0	0	22819.59	-3.09E-02
80.34	0.02	0.34	-0.03	0	0	22819.53	-2.84E-02

Thick Panel (256 plies) Training Set

[Input Ranges]

Tmid	80	500
Tmid-Ttop	-10	50
Tmid-Taut	-100	200
dT/dt	-2.5	5
Hr	0	4

[Output Ranges]

Tadj	-12	12

[Training Data]

Tmid (Y2)	id-Ttop (	id-Taut (	dT/dt (Y1)	Hr (Y1)	Tadj (Y1)	not used	not used
80.00	-0.01	-1.00	0.00	3.28E-03	1	0.19692	8.53E-03
80.01	-0.03	-1.99	0.00	3.28E-03	1	0.393751	3.11E-02
80.01	-0.09	-2.99	0.00	3.26E-03	1	0.589565	5.93E-02
80.01	-0.17	-3.99	0.00	3.21E-03	1	0.781992	8.65E-02
80.02	-0.28	-4.98	0.00	3.09E-03	1	0.967638	0.111426
80.02	-0.41	-5.98	0.00	2.92E-03	1	1.143074	0.134011
80.03	-0.56	-6.97	0.01	2.71E-03	1	1.3056	0.154518
80.03	-0.73	-7.97	0.01	2.47E-03	1	1.453536	0.173236
80.04	-0.91	-9.96	0.01	2.21E-03	2	1.58618	0.196508
80.05	-1.14	-11.95	0.01	1.95E-03	2	1.70345	0.235261
80.07	-1.40	-13.93	0.02	1.69E-03	2	1.804968	0.278227
80.09	-1.70	-15.91	0.02	1.41E-03	2	1.889311	0.319197
80.11	-2.03	-17.89	0.02	1.08E-03	2	1.954299	0.356983
80.14	-2.40	-20.86	0.03	7.26E-04	3	1.997839	0.397759
80.18	-2.81	-23.82	0.04	3.43E-04	3	2.018418	0.452587
80.22	-3.28	-26.78	0.04	0	3	2.014651	0.510386
80.28	-3.79	-29.72	0.05	0	3	1.984655	0.565119
80.34	-4.35	-32.66	0.06	0	3	1.926286	0.615745
80.41	-4.94	-35.59	0.07	0	3	1.837867	0.662476
80.50	-5.56	-39.50	0.09	0	4	1.718714	0.711852
80.60	-6.24	-43.40	0.10	0	4	1.569107	0.775033
80.71	-6.97	-47.29	0.11	0	4	1.389451	0.840969
80.84	-7.74	-51.16	0.13	0	4	1.179459	0.903635
80.99	-8.56	-56.01	0.15	0	5	0.938333	0.968085
81.15	-9.44	-60.85	0.16	0	5	0.665336	1.045252
81.33	-10.38	-65.67	0.18	0	5	0.359677	1.124155
81.54	-11.38	-70.46	0.21	0	5	0.020133	1.198897
81.77	-12.42	-76.23	0.23	0	6	0	1.274656
82.02	-13.53	-81.98	0.25	0	6	0	1.362474
82.30	-14.70	-87.70	0.28	0	6	0	1.45146
82.61	-15.93	-93.39	0.31	0	6	0	1.535791

82.94	-17.22	-100.06	0.34	0	7	0	1.62071
83.31	-18.57	-106.69	0.37	0	7	0	1.717313
83.71	-19.98	-113.29	0.40	0	7	0	1.814755
84.14	-21.46	-119.86	0.43	0	7	0	1.907256
84.61	-22.99	-127.39	0.47	0	8	0	2.000093
85.12	-24.58	-134.88	0.51	0	8	0	2.104396
85.67	-26.25	-142.33	0.55	0	8	0	2.209354
86.26	-27.97	-149.74	0.59	0	8	0	2.309214
86.89	-29.74	-158.11	0.63	0	9	0	2.409281
87.57	-31.58	-166.43	0.68	0	9	0	2.520716
88.30	-33.49	-174.70	0.73	0	9	0	2.63274
89.07	-35.45	-181.93	0.78	0	8	0	2.73354
89.90	-37.43	-189.10	0.83	0	8	0	2.811645
90.78	-39.43	-196.22	0.88	0	8	0	2.878962
91.72	-41.43	-202.28	0.93	0	7	0	2.936446
92.71	-43.42	-208.29	0.99	0	7	0	2.975345
93.76	-45.38	-214.24	1.05	0	7	0	3.007693
94.86	-47.30	-219.14	1.11	0	6	0	3.034009
96.03	-49.18	-223.97	1.17	0	6	0	3.045035
97.26	-51.00	-227.74	1.23	0	5	0	3.046261
98.54	-52.74	-230.46	1.29	0	4	0	3.027083
99.89	-54.38	-232.11	1.35	0	3	0	2.983234
101.29	-55.89	-232.71	1.40	0	2	0	2.915515
102.76	-57.25	-232.24	1.46	0	1	0	2.825777
104.27	-58.46	-231.73	1.52	0	1	0	2.721967
105.85	-59.51	-231.15	1.57	0	1	0	2.622582
107.47	-60.42	-230.53	1.62	0	1	0	2.5345
109.14	-61.21	-229.86	1.67	0	1	0	2.458039
110.85	-61.88	-229.15	1.71	9.65E-04	1	0	2.392033
112.61	-62.46	-227.39	1.76	4.33E-03	0	0	2.329052
114.40	-62.92	-225.60	1.79	7.87E-03	0	0	2.257083
116.23	-63.28	-223.77	1.83	1.16E-02	0	0	2.186473
118.08	-63.55	-221.92	1.86	0.015465	0	0	2.122729
119.96	-63.74	-220.04	1.88	1.96E-02	0	0	2.066468
121.87	-63.85	-218.13	1.91	2.39E-02	0	0	2.017116
123.79	-63.90	-216.21	1.93	0.028433	0	0	1.973916
125.74	-63.89	-214.26	1.94	3.32E-02	0	0	1.936158
127.70	-63.83	-212.30	1.96	0.038335	0	0	1.903225
129.67	-63.73	-210.33	1.97	4.37E-02	0	0	1.874594
131.65	-63.60	-208.35	1.98	0.049479	0	0	1.849821
133.65	-63.43	-206.35	1.99	5.56E-02	0	0	1.828529
135.65	-63.24	-204.35	2.00	6.22E-02	0	0.476559	1.810396
137.66	-63.02	-202.34	2.01	6.93E-02	0	4.63168	1.795147
139.68	-62.79	-200.32	2.02	0.07687	0	9.243903	1.782546
141.71	-62.53	-198.29	2.03	8.51E-02	0	14.35019	1.772391
143.74	-62.27	-196.26	2.03	9.40E-02	0	19.9916	1.764505
145.78	-61.99	-194.22	2.04	0.103701	0	26.21366	1.758739
147.82	-61.70	-192.18	2.05	0.114216	0	33.06662	1.754963

149.88	-61.40	-190.12	2.05	0.125655	0	40.60592	1.753065
151.94	-61.09	-188.06	2.06	0.138111	0	48.89258	1.75295
154.01	-60.77	-185.99	2.07	0.151685	0	57.9937	1.754536
156.09	-60.45	-183.91	2.08	0.166488	0	67.983	1.757755
158.18	-60.12	-181.82	2.09	0.182641	0	78.94147	1.762549
160.28	-59.79	-179.72	2.10	0.200277	0	90.9581	1.76887
162.40	-59.45	-177.60	2.12	0.219543	0	104.1306	1.77668
164.53	-59.10	-175.47	2.13	0.2406	0	118.5666	1.785947
166.69	-58.74	-173.31	2.15	0.263629	0	134.3844	1.796651
168.86	-58.38	-171.14	2.17	0.288831	0	151.7142	1.808773
171.05	-58.01	-168.95	2.20	0.316429	0	170.6999	1.822306
173.27	-57.62	-166.73	2.22	0.346674	0	191.5004	1.837246
175.52	-57.23	-164.48	2.25	0.379848	0	214.2912	1.853596
177.81	-56.82	-162.19	2.28	0.416268	0	239.2673	1.871364
180.12	-56.39	-158.88	2.32	0.456293	-1	266.6448	1.884409
182.48	-55.90	-154.52	2.36	0.500332	-2	296.6648	1.875321
184.88	-55.34	-149.12	2.40	0.548867	-3	329.5968	1.837758
187.33	-54.66	-142.67	2.45	0.602467	-4	365.7448	1.771783
189.83	-53.84	-135.17	2.50	0.661801	-5	405.4528	1.67888
192.39	-52.84	-127.61	2.56	0.727617	-5	449.1099	1.566816
195.02	-51.67	-119.98	2.63	0.800741	-5	497.1543	1.454514
197.71	-50.33	-112.29	2.69	0.882049	-5	550.0773	1.349635
200.48	-48.81	-104.52	2.77	0.972465	-5	608.4251	1.253217
203.33	-47.13	-96.67	2.85	1.072954	-5	672.8024	1.164714
206.26	-45.28	-88.74	2.93	1.184553	-5	743.8755	1.083341
209.28	-43.27	-80.72	3.02	1.308379	-5	822.3783	1.008401
212.40	-41.10	-72.60	3.11	1.445642	-5	909.1168	0.939333
215.61	-38.76	-64.39	3.21	1.597635	-5	1004.975	0.875704
218.93	-36.25	-56.07	3.32	1.76572	-5	1110.918	0.817184
222.36	-33.58	-47.64	3.43	1.951295	-5	1227.996	0.763521
225.92	-30.74	-39.08	3.55	2.155739	-5	1357.34	0.714518
229.60	-27.73	-30.40	3.68	2.380334	-5	1500.16	0.670019
233.42	-24.54	-21.58	3.82	2.626147	-5	1657.729	0.629892
237.38	-21.17	-12.62	3.96	2.89386	-5	1831.361	0.594014
241.49	-17.63	-3.51	4.11	3.183543	-5	2022.373	0.562257
245.75	-13.90	5.75	4.26	3.494345	-5	2232.034	0.534477
250.17	-9.99	15.17	4.42	3.824105	-5	2461.48	0.510493
254.75	-5.91	24.75	4.57	4.168886	-5	2711.613	0.49007
259.46	-1.66	34.46	4.72	4.522462	-5	2982.961	0.4729
264.31	2.73	44.31	4.85	4.87583	-5	3275.511	0.458569
269.27	7.23	53.27	4.96	5.216892	-4	3588.524	0.452668
274.30	11.79	61.30	5.03	5.530478	-3	3920.353	0.471692
279.36	16.33	68.36	5.06	5.798963	-2	4268.291	0.520694
284.39	20.77	75.39	5.03	6.003673	-2	4628.511	0.591936
289.32	25.04	82.32	4.93	6.127162	-2	4996.141	0.665011
294.09	29.07	89.09	4.76	6.156093	-2	5365.506	0.730463
298.61	32.81	95.61	4.52	6.084016	-2	5730.547	0.785287
302.82	36.19	101.82	4.21	5.913154	-2	6085.337	0.828001

306.68	39.19	107.68	3.85	5.654469	-2	6424.605	0.85742
310.13	41.78	114.13	3.45	5.325872	-3	6744.157	0.866345
314.01	44.82	122.01	3.88	6.177387	-4	7114.8	0.837123
318.79	48.46	131.79	4.78	7.070487	-5	7539.03	1.130911
323.97	51.82	142.97	5.19	7.620627	-6	7996.267	1.831113
329.60	55.58	154.60	5.63	8.228855	-6	8489.999	1.867315
335.72	59.80	166.72	6.12	8.911281	-6	9024.676	1.901741
342.41	64.52	179.41	6.69	9.690917	-6	9606.131	1.96929
349.76	69.80	192.76	7.35	10.58329	-6	10241.13	2.069792
357.86	75.70	206.86	8.10	11.60504	-6	10937.43	2.204673
366.83	82.29	221.83	8.97	12.77015	-6	11703.64	2.377815
376.79	89.65	237.79	9.96	14.08702	-6	12548.86	2.595018
387.84	97.84	254.84	11.05	15.54474	-6	13481.54	2.863834
400.07	106.88	273.07	12.23	17.09567	-6	14507.29	3.193571
413.48	116.69	292.48	13.40	18.62063	-6	15624.52	3.595127
427.87	127.00	312.87	14.39	19.8802	-6	16817.34	4.08019
442.75	137.22	333.75	14.89	20.47911	-6	18046.08	4.658996
457.23	146.37	354.23	14.48	19.92137	-6	19241.37	5.335265
470.10	153.14	373.10	12.87	17.86114	-6	20313.03	6.096639
480.25	156.39	389.25	10.15	14.47908	-6	21181.78	6.900302
487.19	155.66	402.19	6.94	10.56344	-6	21815.58	7.658523
491.18	151.42	411.18	4.00	7.029919	-6	22237.38	8.237363
492.94	144.65	412.94	1.76	4.376987	-6	22500	8.534128
493.20	136.43	413.20	0.26	2.619523	-6	22657.17	8.473103
492.52	127.81	412.52	-0.68	1.540987	-6	22749.63	7.941856
491.26	119.59	411.26	-1.26	0.906014	-6	22803.99	6.965108
489.64	112.30	409.64	-1.62	0.53926	-6	22836.35	5.668759
487.79	106.23	407.79	-1.85	0.328623	-6	22856.06	4.217363
485.77	101.45	405.77	-2.02	0.207323	-6	22868.5	2.765263
483.62	97.87	403.62	-2.15	0.13682	-6	22876.71	1.427257
481.37	95.35	401.37	-2.25	9.52E-02	-6	22882.42	0.269702
479.03	93.69	399.03	-2.34	6.99E-02	-6	22886.61	-0.6835
476.60	92.70	396.60	-2.43	5.40E-02	-6	22889.85	-1.43803
474.10	92.22	394.10	-2.50	4.34E-02	-6	22892.46	-2.01597
471.53	92.10	391.53	-2.57	3.61E-02	-6	22894.62	-2.44577
468.90	92.22	388.90	-2.63	3.06E-02	-6	22896.46	-2.75602
466.21	92.50	386.21	-2.69	2.64E-02	-6	22898.04	-2.97235
463.47	92.87	383.47	-2.74	2.29E-02	-6	22899.42	-3.1163
460.68	93.29	380.68	-2.79	2.00E-02	-6	22900.62	-3.20527
457.85	93.71	377.85	-2.83	1.74E-02	-6	22901.66	-3.25299
454.98	94.11	374.98	-2.87	1.51E-02	-6	22902.57	-3.27011
452.08	94.48	372.08	-2.90	1.31E-02	-6	22903.35	-3.26479
449.15	94.79	369.15	-2.93	1.12E-02	-6	22904.02	-3.24326
446.19	95.05	366.19	-2.95	9.40E-03	-6	22904.58	-3.21022
443.22	95.24	363.22	-2.97	7.76E-03	-6	22905.05	-3.16922
440.23	95.37	360.23	-2.99	6.24E-03	-6	22905.42	-3.1229
437.22	95.44	357.22	-3.00	4.82E-03	-6	22905.71	-3.07324
434.21	95.45	354.21	-3.01	3.49E-03	-6	22905.92	-3.0217

431.19	95.40	351.19	-3.02	2.25E-03	-6	22906.06	-2.96935
428.16	95.29	348.16	-3.03	0.001104	-6	22906.12	-2.91696
425.14	95.13	345.14	-3.03	3.70E-05	-6	22906.13	-2.86508
422.11	94.92	342.11	-3.03	0	-6	22906.07	-2.81411
419.09	94.66	339.09	-3.02	0	-6	22905.96	-2.76429
416.07	94.36	336.07	-3.02	0	-6	22905.79	-2.71581
413.06	94.02	333.06	-3.01	0	-6	22905.59	-2.66875
410.06	93.64	330.06	-3.00	0	-6	22905.34	-2.62318
407.07	93.23	327.07	-2.99	0	-6	22905.05	-2.5791
404.09	92.79	324.09	-2.98	0	-6	22904.73	-2.53649
401.13	92.32	321.13	-2.96	0	-6	22904.37	-2.49534
398.18	91.82	318.18	-2.95	0	-6	22903.99	-2.45559
395.24	91.30	315.24	-2.93	0	-6	22903.58	-2.41719
392.32	90.77	312.32	-2.92	0	-6	22903.15	-2.38008
389.42	90.21	309.42	-2.90	0	-6	22902.7	-2.34421
386.54	89.64	306.54	-2.88	0	-6	22902.23	-2.3095
383.68	89.05	303.68	-2.86	0	-6	22901.75	-2.27591
380.84	88.45	300.84	-2.84	0	-6	22901.25	-2.24337
378.01	87.84	298.01	-2.82	0	-6	22900.74	-2.21183
375.21	87.22	295.21	-2.80	0	-6	22900.22	-2.18123
372.43	86.59	292.43	-2.78	0	-6	22899.69	-2.15151
369.67	85.95	289.67	-2.76	0	-6	22899.15	-2.12264
366.93	85.31	286.93	-2.74	0	-6	22898.6	-2.09456
364.21	84.66	284.21	-2.72	0	-6	22898.05	-2.06722
361.52	84.00	281.52	-2.69	0	-6	22897.5	-2.0406
358.85	83.35	278.85	-2.67	0	-6	22896.94	-2.01465
356.20	82.69	276.20	-2.65	0	-6	22896.38	-1.98934
353.57	82.02	273.57	-2.63	0	-6	22895.82	-1.96463
350.96	81.36	270.96	-2.61	0	-6	22895.26	-1.9405
348.38	80.69	268.38	-2.58	0	-6	22894.7	-1.9169
345.82	80.03	265.82	-2.56	0	-6	22894.13	-1.89383
343.28	79.36	263.28	-2.54	0	-6	22893.57	-1.87124
340.77	78.69	260.77	-2.52	0	-6	22893.01	-1.84913
338.28	78.03	258.28	-2.49	0	-6	22892.45	-1.82746
335.81	77.36	255.81	-2.47	0	-6	22891.89	-1.80623
333.36	76.70	253.36	-2.45	0	-6	22891.33	-1.7854
330.93	76.04	250.93	-2.43	0	-6	22890.78	-1.76496
328.53	75.38	248.53	-2.40	0	-6	22890.23	-1.7449
326.15	74.73	246.15	-2.38	0	-6	22889.68	-1.7252
323.79	74.07	243.79	-2.36	0	-6	22889.13	-1.70584
321.45	73.42	241.45	-2.34	0	-6	22888.59	-1.68682
319.13	72.77	239.13	-2.32	0	-6	22888.05	-1.66812
316.84	72.13	236.84	-2.29	0	-6	22887.52	-1.64973
314.57	71.49	234.57	-2.27	0	-6	22886.98	-1.63164
312.31	70.85	232.31	-2.25	0	-6	22886.46	-1.61384
310.08	70.21	230.08	-2.23	0	-6	22885.93	-1.59631
307.87	69.58	227.87	-2.21	0	-6	22885.41	-1.57905
305.68	68.96	225.68	-2.19	0	-6	22884.89	-1.56206

303.51	68.33	223.51	-2.17	0	-6	22884.38	-1.54532
301.37	67.71	221.37	-2.15	0	-6	22883.87	-1.52883
299.24	67.10	219.24	-2.13	0	-6	22883.37	-1.51257
297.13	66.49	217.13	-2.11	0	-6	22882.87	-1.49654
295.04	65.88	215.04	-2.09	0	-6	22882.37	-1.48075
292.97	65.28	212.97	-2.07	0	-6	22881.88	-1.46517
290.93	64.68	210.93	-2.05	0	-6	22881.39	-1.4498
288.90	64.08	208.90	-2.03	0	-6	22880.91	-1.43464
286.89	63.49	206.89	-2.01	0	-6	22880.43	-1.41969
284.90	62.91	204.90	-1.99	0	-6	22879.95	-1.40493
282.92	62.33	202.92	-1.97	0	-6	22879.48	-1.39037
280.97	61.75	200.97	-1.95	0	-6	22879.01	-1.37599
279.04	61.18	199.04	-1.93	0	-6	22878.55	-1.3618
277.12	60.61	197.12	-1.92	0	-6	22878.09	-1.34779
275.22	60.04	195.22	-1.90	0	-6	22877.63	-1.33396
273.34	59.49	193.34	-1.88	0	-6	22877.18	-1.3203
271.48	58.93	191.48	-1.86	0	-6	22876.74	-1.30681
269.64	58.38	189.64	-1.84	0	-6	22876.29	-1.29348
267.81	57.83	187.81	-1.83	0	-6	22875.86	-1.28032
266.01	57.29	186.01	-1.81	0	-6	22875.42	-1.26731
264.21	56.76	184.21	-1.79	0	-6	22874.99	-1.25446
262.44	56.22	182.44	-1.77	0	-6	22874.56	-1.24177
260.68	55.70	180.68	-1.76	0	-6	22874.14	-1.22922
258.94	55.17	178.94	-1.74	0	-6	22873.72	-1.21682
257.22	54.65	177.22	-1.72	0	-6	22873.31	-1.20457
255.51	54.14	175.51	-1.71	0	-6	22872.9	-1.19245
253.82	53.63	173.82	-1.69	0	-6	22872.49	-1.18048
252.15	53.12	172.15	-1.67	0	-6	22872.09	-1.16864
250.49	52.62	170.49	-1.66	0	-6	22871.69	-1.15694
248.85	52.13	168.85	-1.64	0	-6	22871.29	-1.14537
247.22	51.63	167.22	-1.63	0	-6	22870.9	-1.13393
245.61	51.15	165.61	-1.61	0	-6	22870.51	-1.12262
244.01	50.66	164.01	-1.60	0	-6	22870.13	-1.11144
242.43	50.18	162.43	-1.58	0	-6	22869.75	-1.10038
240.87	49.71	160.87	-1.57	0	-6	22869.37	-1.08944
239.32	49.23	159.32	-1.55	0	-6	22869	-1.07862
237.78	48.77	157.78	-1.54	0	-6	22868.63	-1.06792
236.26	48.30	156.26	-1.52	0	-6	22868.26	-1.05734
234.76	47.85	154.76	-1.51	0	-6	22867.89	-1.04687
233.27	47.39	153.27	-1.49	0	-6	22867.54	-1.03652
231.79	46.94	151.79	-1.48	0	-6	22867.18	-1.02627
230.33	46.49	150.33	-1.46	0	-6	22866.83	-1.01614
228.88	46.05	148.88	-1.45	0	-6	22866.48	-1.00611
227.44	45.61	147.44	-1.43	0	-6	22866.13	-0.9962
226.02	45.18	146.02	-1.42	0	-6	22865.79	-0.98638
224.62	44.75	144.62	-1.41	0	-6	22865.45	-0.97667
223.22	44.32	143.22	-1.39	0	-6	22865.11	-0.96707
221.84	43.90	141.84	-1.38	0	-6	22864.78	-0.95756

220.47	43.48	140.47	-1.37	0	-6	22864.45	-0.94815
219.12	43.06	139.12	-1.35	0	-6	22864.12	-0.93885
217.78	42.65	137.78	-1.34	0	-6	22863.8	-0.92964
216.45	42.25	136.45	-1.33	0	-6	22863.48	-0.92052
215.14	41.84	135.14	-1.32	0	-6	22863.16	-0.9115
213.83	41.44	133.83	-1.30	0	-6	22862.84	-0.90257
212.54	41.05	132.54	-1.29	0	-6	22862.53	-0.89374
211.27	40.65	131.27	-1.28	0	-6	22862.22	-0.885
210.00	40.26	130.00	-1.27	0	-6	22861.92	-0.87634
208.75	39.88	128.75	-1.25	0	-6	22861.62	-0.86778
207.51	39.50	127.51	-1.24	0	-6	22861.32	-0.8593
206.28	39.12	126.28	-1.23	0	-6	22861.02	-0.85091
205.06	38.75	125.06	-1.22	0	-6	22860.72	-0.8426
203.86	38.37	123.86	-1.21	0	-6	22860.43	-0.83438
202.66	38.01	122.66	-1.19	0	-6	22860.14	-0.82625
201.48	37.64	121.48	-1.18	0	-6	22859.86	-0.81819
200.31	37.28	120.31	-1.17	0	-6	22859.58	-0.81022
199.15	36.92	119.15	-1.16	0	-6	22859.29	-0.80233
198.00	36.57	118.00	-1.15	0	-6	22859.02	-0.79451
196.86	36.22	116.86	-1.14	0	-6	22858.74	-0.78678
195.74	35.87	115.74	-1.13	0	-6	22858.47	-0.77912
194.62	35.53	114.62	-1.12	0	-6	22858.2	-0.77154
193.52	35.19	113.52	-1.10	0	-6	22857.93	-0.76404
192.42	34.85	112.42	-1.09	0	-6	22857.67	-0.75661
191.34	34.52	111.34	-1.08	0	-6	22857.41	-0.74926
190.27	34.18	110.27	-1.07	0	-6	22857.15	-0.74198
189.20	33.86	109.20	-1.06	0	-6	22856.89	-0.73477
188.15	33.53	108.15	-1.05	0	-6	22856.64	-0.72763
187.11	33.21	107.11	-1.04	0	-6	22856.38	-0.72057
186.07	32.89	106.07	-1.03	0	-6	22856.13	-0.71357
185.05	32.57	105.05	-1.02	0	-6	22855.89	-0.70665
184.04	32.26	104.04	-1.01	0	-6	22855.64	-0.69979
183.04	31.95	103.04	-1.00	0	-6	22855.4	-0.693
182.04	31.64	102.04	-0.99	0	-5	22855.16	-0.68628
181.06	31.34	101.06	-0.98	0	-4	22854.92	-0.67963
180.09	31.04	100.09	-0.97	0	-3	22854.69	-0.67304
179.12	30.74	99.12	-0.96	0	-2	22854.45	-0.66652
178.16	30.45	98.16	-0.96	0	-1	22854.22	-0.66006
177.22	30.15	97.22	-0.95	0	0	22853.99	-0.65366
176.28	29.86	95.28	-0.94	0	1	22853.77	-0.64124
175.35	29.55	94.35	-0.93	0	0	22853.54	-0.61817
174.43	29.23	93.43	-0.92	0	0	22853.32	-0.60661
173.52	28.93	92.52	-0.91	0	0	22853.1	-0.60147
172.62	28.62	91.62	-0.90	0	0	22852.87	-0.59771
171.73	28.32	90.73	-0.89	0	0	22852.64	-0.59401
170.85	28.03	89.85	-0.88	0	0	22852.42	-0.59012
169.97	27.74	88.97	-0.87	0	0	22852.19	-0.586
169.11	27.46	88.11	-0.87	0	0	22851.96	-0.58168

168.25	27.18	87.25	-0.86	0	0	22851.74	-0.5772
167.40	26.90	86.40	-0.85	0	0	22851.52	-0.5726
166.57	26.63	85.57	-0.84	0	0	22851.3	-0.56791
165.74	26.37	84.74	-0.83	0	0	22851.09	-0.56315
164.91	26.10	83.91	-0.82	0	0	22850.88	-0.55834
164.10	25.85	83.10	-0.81	0	0	22850.67	-0.5535
163.30	25.60	83.30	-0.80	0	-1	22850.47	-0.55472
162.50	25.37	82.50	-0.80	0	-1	22850.26	-0.56664
161.71	25.14	81.71	-0.79	0	-1	22850.07	-0.56711
160.93	24.92	80.93	-0.78	0	-1	22849.88	-0.56123
160.16	24.71	80.16	-0.77	0	-1	22849.69	-0.55404
159.39	24.49	79.39	-0.77	0	-1	22849.51	-0.54684
158.63	24.27	78.63	-0.76	0	0	22849.33	-0.53993
157.88	24.05	77.88	-0.75	0	0	22849.16	-0.53333
157.14	23.83	77.14	-0.74	0	0	22848.99	-0.527
156.40	23.61	76.40	-0.74	0	0	22848.83	-0.52092
155.67	23.40	75.67	-0.73	0	0	22848.66	-0.51504
154.94	23.18	74.94	-0.72	0	0	22848.5	-0.50934
154.22	22.97	74.22	-0.72	0	0	22848.34	-0.5038
153.51	22.75	73.51	-0.71	0	0	22848.17	-0.49839
152.81	22.54	72.81	-0.71	0	0	22848.01	-0.49311
152.11	22.33	72.11	-0.70	0	0	22847.85	-0.48794
151.42	22.12	71.42	-0.69	0	0	22847.69	-0.48287
150.73	21.91	70.73	-0.69	0	0	22847.53	-0.47789
150.05	21.71	70.05	-0.68	0	0	22847.38	-0.473
149.38	21.50	69.38	-0.67	0	0	22847.22	-0.46819
148.71	21.30	68.71	-0.67	0	0	22847.06	-0.46345
148.05	21.09	68.05	-0.66	0	0	22846.91	-0.45878
147.39	20.89	67.39	-0.65	0	0	22846.75	-0.45419
146.74	20.70	66.74	-0.65	0	0	22846.6	-0.44965
146.10	20.50	66.10	-0.64	0	0	22846.45	-0.44518
145.47	20.30	65.47	-0.64	0	0	22846.29	-0.44077
144.83	20.11	64.83	-0.63	0	0	22846.14	-0.43641
144.21	19.92	64.21	-0.62	0	0	22845.99	-0.43211
143.59	19.73	63.59	-0.62	0	0	22845.85	-0.42786
142.98	19.54	62.98	-0.61	0	0	22845.7	-0.42366
142.37	19.35	62.37	-0.61	0	0	22845.55	-0.4195
141.77	19.16	61.77	-0.60	0	0	22845.41	-0.4154
141.18	18.98	61.18	-0.60	0	0	22845.27	-0.41134
140.59	18.80	60.59	-0.59	0	0	22845.13	-0.40733
140.00	18.62	60.00	-0.58	0	0	22844.99	-0.40336
139.42	18.44	59.42	-0.58	0	0	22844.85	-0.39943
138.85	18.26	58.85	-0.57	0	0	22844.71	-0.39554
138.28	18.09	58.28	-0.57	0	0	22844.57	-0.3917
137.72	17.91	57.72	-0.56	0	0	22844.44	-0.38789
137.17	17.74	57.17	-0.56	0	0	22844.3	-0.38413
136.62	17.57	56.62	-0.55	0	0	22844.17	-0.3804
136.07	17.40	56.07	-0.55	0	0	22844.04	-0.37671

135.53	17.23	55.53	-0.54	0	0	22843.91	-0.37306
134.99	17.07	54.99	-0.54	0	0	22843.78	-0.36945
134.46	16.90	54.46	-0.53	0	0	22843.65	-0.36587
133.94	16.74	53.94	-0.53	0	0	22843.52	-0.36233
133.42	16.58	53.42	-0.52	0	0	22843.4	-0.35883
132.90	16.42	52.90	-0.51	0	0	22843.27	-0.35536
132.39	16.26	52.39	-0.51	0	0	22843.15	-0.35192
131.89	16.10	51.89	-0.51	0	0	22843.03	-0.34852
131.39	15.95	51.39	-0.50	0	0	22842.91	-0.34515
130.89	15.79	50.89	-0.50	0	0	22842.79	-0.34181

APPENDIX C

Physical Properties Used
Panel Thicknesses Used

Table C.1

Physical Properties Used

Properties of Hercules [38]

(Initial prepreg resin mass fraction = 0.42)

Property	Resin	Fiber
Thermal Conductivity (W/mK)	0.1670	26.000
Density kg/m^3	1260.0	1790.0
Specific Heat Capacity kJ/(kg.K)	1.26	0.712
Heat of Reaction J/g	474	

The properties of Hercules AS4/3501-6 Prepreg and Mochburg CW 1850 Thermal Fiber Bleeder Cloth shown in Table C.1 were used in the simulator calculations.

Table C.2

Panel Thicknesses Used

Layer	# of nodes	32-ply		128-ply		256-ply	
		# of plies	thickness	# of plies	thickness	# of plies	thickness
breather	1	1	0.01292	1	0.01292	1	0.01292
bleeder	2	3	0.00153	12	0.00612	24	0.01224
laminate	16	32	0.00173	128	0.00693	256	0.01387
bleeder	2	3	0.00153	12	0.00612	24	0.01224
breather	1	1	0.01292	1	0.01292	1	0.01292